VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF CONTROL AND INSTRUMENTATION

SROVNÁNÍ FRAMEWORKŮ PRO VÝVOJ DATABÁZOVÝCH APLIKACÍ

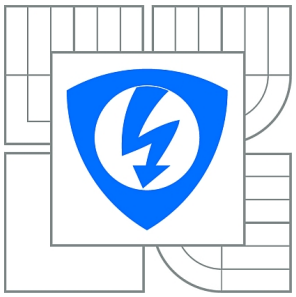
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. MARTIN DUFEK
AUTHOR

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV AUTOMATIZACE A MĚŘICÍ TECHNIKY

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF CONTROL AND INSTRUMENTATION

SROVNÁNÍ FRAMEWORKŮ PRO VÝVOJ DATABÁZOVÝCH APLIKACÍ

THE COMPARATION OF THE FRAMEWORK TOOLS

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

AUTOR PRÁCE — Bc. MARTIN DUFEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. RADOVAN HOLEK, CSc.

SUPERVISOR

BRNO 2013

VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav automatizace a měřicí techniky

Diplomová práce

magisterský navazující studijní obor
Kybernetika, automatizace a měření

Student: Bc. Martin Dufek
Ročník: 2

ID: 106414
Akademický rok: 2012/13

NÁZEV TÉMATU:

Srovnání frameworků pro vývoj databázových aplikací

POKYNY PRO VYPRACOVÁNÍ:

1. Seznamte se s dostupnými vývojovými nástroji (frameworky) pro podporu tvorby databázových aplikací, které budou provozované v prostředí tenkého klienta(www).
2. Porovnejte možnosti, výhody a nevýhody dvou vybraných vývojových prostředí při tvorbě aplikací.
3. Podrobně zmapujte knihovny, které jednotlivé vývojové nástroje používají. Popište význam knihoven, jejich obsah a použití při návrhu. Zmapujte volně dostupné knihovny k těmto vývojovým prostředím.
4. Navrhněte dvě datově a procesně shodné vzorové aplikace pomocí obou vývojových nástrojů (frameworků). Aplikace budou obsahovat minimálně 10 tabulek, budou obsahovat náležitosti pro správu uživatelů a přidělování rolí jednotlivým uživatelům, nástroje pro vyhledávání, filtrování, stránkování, ošetření správnosti vstupních dat a práci s elektronickou poštou.
5. Vzorové aplikace budou také řešit životní cyklus vybraných entit v sovislosti s přidělenými rolemi jednotlivým úživatelům.
6. Porovnejte časovou a znalostí složitost návrhu aplikací v obou prostředích. Zhodnoťte obtížnost pro naučení a práci s nástroji.

DOPORUČENÁ LITERATURA:

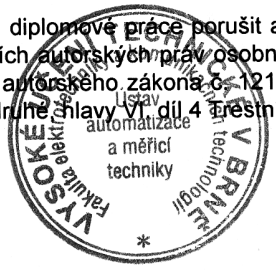
Maslakowski M., Naučte se MySQL za 21 dní. Praha: Computer Press, 2001. 478 s. ISBN 80-72226-448-6.
Dle pokynů vedoucího práce a vlastního výběru.

Termín zadání: 11.2.2013 ***Termín odevzdání:*** 20.5.2013

Vedoucí práce: Ing. Radovan Holek, CSc.
Konzultanti diplomové práce:

doc. Ing. Václav Jirsík, CSc.
předseda oborové rady

UPOZORNĚNÍ:



ABSTRAKT

Tato diplomová práce se zabývá seznámením se s dostupnými frameworky v jazyce php, pracujícím v prostředí tenkého klienta. Zaměřuje se na porovnání dvou frameworků CodeIgniter a Zend Framework. V další části nabízí podrobný popis knihoven frameworků. Pro porovnání jsem navrhl oběma frameworky vzorovou aplikaci, obsahující základní prvky, které se využívající při tvorbě většiny webových aplikací. V závěru jsou porovnány výhody a nevýhody vývojových prostředí.

KLÍČOVÁ SLOVA

Framework, nástroje pro tenkého klienta, model, pohled, kontroler, PHP, SQL, Zend framework, CodeIgniter

ABSTRACT

This diploma thesis deals with familiarization with available frameworks in the PHP language working in a thin client environment. It focuses on the comparison of two frameworks CodeIgniter and Zend Framework. The next section provides a detailed description of framework libraries. I suggested sample application made by two frameworks for a comparison, which contains the basic elements, that are used in creating majority of web applications. In the conclusion there are compared pros and cons of development environments.

KEYWORDS

Framework, tools for thin client, model, view, controller, PHP, SQL, Zend framework, CodeIgniter

DUFEK, Martin *Srovnání frameworků pro vývoj databázových aplikací*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav automatizace a měřicí techniky, 2013. 85 s. Vedoucí práce byl Ing. Radovan Holek, CSc.

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Srovnání frameworků pro vývoj databázových aplikací“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

Poděkování
Děkuji vedoucímu diplomové práce Ing. Radovan Holek, CSc. za účinnou metodickou, pedagogickou a odbornou pomoc a další cenné rady při zpracování mé semestrální práce. Děkuji rodičům za podporu.

V Brně dne ..... Podpis autora .....

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Je plno volně dostupných frameworků, pro tvorbu webových stránek v jazyce PHP. V této práci se budu zabývat jejich zmapováním. V další části se zaměřím na podrobné prostudování a zdokumentování nabízených knihoven od dvou zvolených frameworků. Dále porovnám vlastnosti vybraných frameworků Zend framework a CodeIgniter, porovnám nabídku knihoven, výhody, nedostatky, dostupnost a přehlednost dokumentace. V další části této práce navrhnu shodné aplikace pomocí vybraných nástrojů. Aplikace navrhnu tak, aby splňovaly základní požadavky webových aplikací. Ze zkušeností při návrhu porovnám výhody a nevýhody vytvořených aplikací. V závěru zhodnotím vlastnosti obou frameworků a určím, který je lepší. Zhodnotím náročnost na naučení a práci s frameworky.

1 VÝVOJOVÉ NÁSTROJE

1.1 PHP

PHP[1] je skriptovací jazyk vykonávaný na straně serveru, který se vkládá do běžného HTML kódu. Oproti HTML se liší v tom, že server nejprve vezme skript PHP a vykoná všechny příkazy v PHP, které jsou na stránce uvedeny, následně pošle klientovi čistý HTML kód. V současné době se jazyk PHP nachází ve verzi 5. Pomocí PHP lze ukládat data od klientů do textových souborů, nebo do databáze(například mySQL). Je to volně šiřitelný programovací jazyk poskytovaný zadarmo.

1.2 SQL

(strukturovaný dotazovací jazyk)[3] je standardním jazykem pro komunikaci s relačními databázemi. Tento jazyk, coby nástroj pro vytváření databází, jejich správu, zabezpečení a dotazování podporují téměř všechny relační databázové systémy. Je to plnohodnotný jazyk pro veškeré práce v databázích.

1.3 Framework

Framework[4] je softwarová struktura, která slouží jako podpora při programování, vývoji a organizaci jiných softwarových projektů. Může obsahovat podpůrné programy, knihovny API, návrhové vzory nebo doporučené postupy při vývoji. PHP frameworky se skládají ze sad skriptů v PHP, které pracují a fungují jako ty naše. Nejedná se o žádné kompilované knihovny, které bychom museli do PHP přidávat.

1.3.1 Zend framework

Zend framework[5] je objektově orientovaný webový framework prazující v jazyce PHP, zkráceně označovaný jako ZF. ZF je vyvíjen s ohledem na jednoduchý vývoj webových aplikací. Užívá modulární architekturu, která umožňuje vývojářům použít jen ty komponenty, které potřebují. ZF nabízí dvě možnosti inicializace, buď pomocí souboru INI nebo XML. Zahrnuje v sobě komponenty pro MVC aplikace. Umožňuje spolupráci s databázovými systémy MySQL, MSSQL, Oracle, PostgreSQL a IBM DB2. Projekty ZF jsou tvořeny kombinací jazyků PHP a HTML mající příponu phtml. Nabízí validační metody pro kontrolu správného formátu datumu, mailu, IP adres, čísel a dalších. Obsahuje helpery pro usnadnění stále se opakujících kódů. Pro ZF jsou dostupné API od výrobců jako je Google, Amazon, Yahoo!

a Flickr. Nabízí tvorbu a zpracování RSS nebo správu uživatelů přes OpenID. Má knihovny pro práci s maily, pro komunikaci s jinými webovými servery a pro tvorbu pdf souborů. ZF má na svých stránkách zpracovanou a přehlednou dokumentaci všech svých knihoven, stránku s tutoriály a forum s řešenými problémy. Licence je bezplatná nebo za dobrovolný poplatek.

1.3.2 CakePHP

CakePHP[6] je zdarma šiřitelný open-source pro PHP. Aktuální verze je CakePHP2.3.2. Je postaven na základě Ruby on Rails. Je kompatibilní s PHP ve verzi 4 i 5. Nabízí automatické generování kódu, rychlé a flexibilní šablony. CakePHP je navržen pro maximální efektivitu práce. Ve skutečnosti stačí napsat určitý kód pouze jednou, a pak ho používat podle potřeby. Dále pak šablony pro AJAX, JavaScript, HTML formuláře a další. Funkce pro práci s Emaily, Cookies, zabezpečením a sekcemi. Podporuje databázový systémy DB2, Firebird, MSSQL, MySQL, ODBC, Oracle, PostgreSQL, SQLite. Funguje na jakémkoliv hostingu s minimální nebo žádnou konfigurací.

1.3.3 Symfony

Symfony[7] je aplikační framework psaný v jazyce PHP,minimální verze PHP pro Symfony je PHP5.2.4. Aktuální verze je Symfony2.2.1. Je založena na architektuře MVC. Obsahuje řadu nástrojů a tříd zaměřených na zkrácení doby vývoje webových aplikací. Symfony nabízí knihovny pro podporu:

- AJAX
- správu cache
- informace o toku dat
- šablony a helpery
- možnost pluginů

Symfony stačí na server nahrát pouze jednou do adresáře, který není přístupný přes HTTP. Mnoho funkcí lze volat přes příkazový řádek. Nastavení aplikace využívá kaskádový přístup, nejvyšší nastavení může být překryto podle nastavení projektu. Symfony lze nainstalovat na všechny hlavní operační systémy a je kompatibilní s většinou databází.

1.3.4 Ruby On Rails

Ruby On Rails[8] je framework pro vývoj webových aplikacích v jazyce Ruby, zkráceně označován jako Rails. Jeho základem je MVC. Umožňuje psát méně kódu a zároveň dosáhnout více než v mnoha jiných jazycích. Rails nabízí řadu knihoven:

- Action Mailer - pro práci s e-maily, umožňuje odesílat e-maily i s přílohami, na základě mnoha šablon. Také umí emaily přijímat a zpracovávat.
- Action Dispatch - zpracovává příchozí požadavky a posílá je, kam chcete.
- Action View - má na starosti zobrazování ve formátu HTML a XML.

1.3.5 Nette

Nette framework[9] je výkonný framework pro pohodlné a rychlé vytváření kvalitních webových aplikacích v PHP. Nette vyžaduje minimálně PHP řady 5.2.0. Podporuje AJAX, SEO, DRY, KISS, MVC a znovupoužitelnost kódu. Nabízí velkou řadu rozšiřujících doplňků pro snadnější používání. Nette jako jediný z těchto frameworků nabízí české stránky s podrobnou dokumentací a příručkou pro programátora s postupy a návody. Na oficiálních stránkách je také fórum.

1.3.6 CodeIgniter

CodeIgniter[10] je framework založený na principu MVC pro vývoj webových aplikací v PHP. Aktuální verzí je CodeIgniter 2.1.3. Řadí se mezi menší frameworky, samotná velikost je v řádu několika MB. Jednoduchý na instalaci a používání. Snaží se co nejjednodušeji řešit problémy s co nejmenšími nároky na programátora a systém. CodeIgniter běží na základě několika malých knihoven, ostatní knihovny jsou nahrávány dle potřeby. CodeIgniter je vybaven knihovnami pro nahrávání souborů, pro práci s maily, dále obsahuje knihovnu pro zpracování obrázku, jako je například zmenšování, vkládání vodoznaku a ořez. Nabízí řadu helperů, které slouží k zpřehlednění a nahrazení stále se opakujících částí kódu. U CodeIgniter je struktura programu rozdělena do dvou částí:

- Systém - v této složce jsou uloženy všechny knihovny a helpery.
- Application - je část pro samotnou aplikaci a všechny součásti tvořené uživatelem. Zde je možno vkládat vlastní nebo upravený kód ze systémových knihoven či helperů s příponou "my_ ". CodeIgniter sám zjistí, zda chcete používat upravenou systémovou knihovnu a přednostně ji použije.

CodeIgniter má zpracovanou podrobnou dokumentaci, na oficiálních stránkách nabízí základní informace, návody, tutoriály, popisy helperů a knihoven s ukázkami kódu.

1.3.7 Prado

Prado[11] je sada komponent pro PHP 5 připomínající koncept, který můžeme nalézt v Borland Delphi. Pro svoji funkčnost potřebuje PHP 5 s XML, veškerá konfigurace aplikace se realizuje přes XML soubory. Je paměťově a procesně náročný.

1.3.8 Jelix

Framework[12] je určen pro PHP5.2 a vyšší. Jedná se o framework, u kterého je třeba používat příkazový řadek k vytvoření aplikace, modulů a objektů. Podporuje více výstupních formátů, kromě XHTML také nabízí XUL, RSS, ATOM, XML, PDF a další. Do výbavy Jelixu patří také jAcl. Jelix neobsahuje Helpery. Formuláře se ukládají jako XML soubory.

1.3.9 Lamplighter

Je[12] určen pro PHP5 a vyšší. Podporuje databázi MySQL. Kromě standardních pluginů obsahuje také plugin Photo/Image Management, který umožňuje různé operace s obrázky. Obsahuje Helpery. Samotná konfigurace a instalace se provádí v příkazové řádce.

1.4 Návrhový vzor MVC

Model view Controller[2] je architektura využívaná u většiny vývojových prostředí pro tvorbu webových aplikací. MVC rozděluje celou aplikaci do tří základních souborových typů, na Model sloužící ke zpracování dat, View prezentace dat a Controller (řadič) řízení. MVC vzájemnou komunikací a výměnou dat tvoří celou aplikaci.

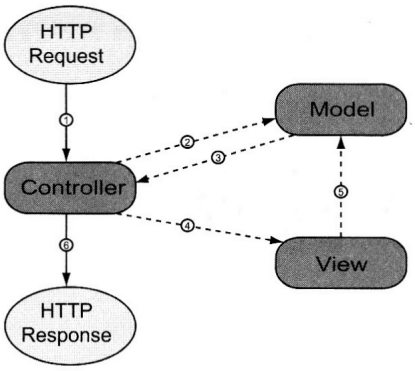
1.4.1 Princip MVC

Na obrázku1.1 je zobrazen HTTP požadavek(HTTP Request) na server. Zpracování požadavku pomocí MVC a zpětná HTTP odpověď(HTTP Response).

1. Vše začíná posláním HTTP požadavku na server. Tento požadavek je přijat řadičem a dále zpracováván.
2. Přijatý požadavek řadič zpracuje, zvolí vhodný model a podle potřeby přepošle žádost o zpracování dat na model.
3. Model zpracuje požadavek ke zpracování dat v databázi. Výsledek své práce vrací řadiči.
4. Řadič upraví data do správného formátu, vybere vhodné zobrazení a odešle je na View.
5. V některých případech se může stát, že si view vyžádá data přímo od modelu.
6. Na závěr řadič vytvoří view a odešle view zpět do prohlížeče ve formě HTTP odpovědi.

Většina programátorů bodu č.5 nevyužívá. Někteří programátoři jsou toho názoru, že view by nemělo mít vůbec přímo přístup do databáze. Ve většině případech si

v aplikacích všechna data vyžádají řadičem od modelu a následně vše předají view, view dále nemá potřebu do modelu zasahovat.



Obr. 1.1: Průběh požadavku MVC[2]

MVC si lze také představit jako tři vrstvy. Vrstva prezenční, ve které se view stará o zobrazení dat uživateli. Vrstva řídící, kterou představuje řadič a stará se o chod celé aplikace. Vrstva datová, která přistupuje k datům. Každá vrstva se stará o to své, dohromady tvoří přehledný celek.

1.4.2 Model

Model[2] se stará o datovou vrstvu v MVC. Umožňuje čtení, zapisování, upravování a mazání dat. O tom, kam data zapíše to ví jenom model. Model umožňuje práci s daty pro různá datová úložiště:

- Databáze
- Textový soubor
- RSS
- Webová služba

Při tvorbě modelu je důležité vědět, jak k datům v určitých úložištích přistupovat. U textových dokumentů je potřeba vědět, jak data formátovat, aby byly srozumitelné pro další použití.

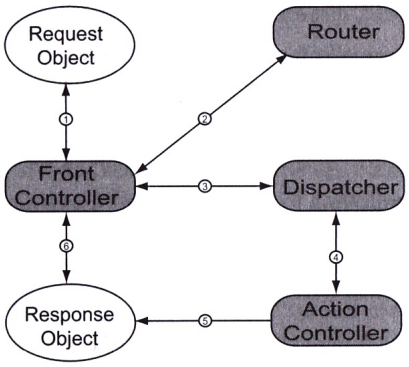
1.4.3 View

Pohled[2] se stará o prezentaci předaných dat uživateli. View se nestará, odkud data jsou, nebo zda jsou správná. View se stará pouze o vložení přijatých dat do zvolené

šablony, správně je naformátovat a publikovat ve srozumitelném a přehledném zobrazení uživateli. Zobrazení musí vědět, v jakém formátu data dostane, zda se jedná o pole, ve kterém budou všechny data, nebo zda budou data předána po prvcích. Zpětně se taky view stará o přijetí dat skrze formuláře. Výpis pohledu se většinou realizuje ve formátu HTML stránky, která je odeslána do prohlížeče uživatele. View umožňuje vytvořit formáty CSV nebo XML, které si uživatel může stáhnout. Dále může view vytvářet grafiku nebo PDF, které se zobrazí v prohlížeči. Všechny formáty jsou do prohlížeče odeslány jako odpověď HTTP.

1.4.4 Controller

Controller[2] představuje řídící vrstvu. Zavolání controlleru se vyvolá HTTP žádostí z prohlížeče uživatele. Z přijaté žádosti controller zjistí, který model má být použit a v jakém view mají být data následně vykreslena. Controller se skládá z více častí, které vkládají různé návrhové vzory. Jsou to front controller a action controller. Na obrázku1.2 je zobrazen průběh požadavku v controlleru. Hlavní částí controlleru je front controller.



Obr. 1.2: Průběh požadavku v controlleru[2]

1. Přijetí požadavku uživatele zařizuje front controller.
2. V dalším kroku front controller předá data Router(směrovači), který zjistí, jaká akce nebo jaké akce mají být vykonány.
3. Následně je front controller pověřen dispečerem, aby vykonal další zpracování požadavku.
4. V cyklu volá dispečer postupně action controllery, aby byly vykonány všechny akce. V určitých akcích může controller přistupovat k modelu.
5. Výsledek je předán responze objektu.
6. V závěru front controller odešle odpověď do prohlížeče uživatele ve formě HTTP odpovědi.

2 KNIHOVNY

2.1 Vlastní knihovny

Jsou knihovny, které jsou součástí frameworku.

2.1.1 CodeIgniter

Calendar[14] - je knihovna s funkcemi pro tvorbu kalendářů. Knihovna nabízí formátování kalendářů pomocí šablon, které zaručují kontrolu nad formátem jeho vzhledu. U vytvořeného kalendáře je možno přistupovat k datům v buňkách kalendáře.
Funkce:
generate - vstupem je rok, měsíc nebo pole, ve kterým jsou oba údaje. Zpracuje vstupní data. Podle počátečního nastavení se nastaví den, kterým začíná týden. Vytvoří hlavičku kalendáře s rokem, názvem aktuálního měsíce a odkazy na předchozí a následující měsíc. Výstupem vrací pole obsahující údaje k zobrazení kalendáře.
get_month_name - vstupem je číslo měsíce a návratovou hodnotou je buď celý název měsíce nebo zkrácený název.
get_day_names - vstupní hodnotou je typ zobrazení názvu dnů. Výstupem je pole názvů dnů. Na výběr je ze tří možností long, short, default (monday, mon, mo).
adjust_date - vstupem je měsíc a rok. V případě špatně zadaného čísla měsíce funkce přepočte měsíc na reálnou hodnotu a upraví rok. Návratovou hodnotou je pole obsahující rok a měsíc.
get_total_days - vstupem je rok a měsíc. Funkce vrací počet dní v měsíci, počítá i s přestupným rokem.
default_template - nastaví základní parametry šablony.
parse_template - slouží k sestavení šablony.

Cart[14] - je knihovna starající se o uchování informací uživatel po dobu otevření stránky. Využívá se pro správu typu nákupní košík. Pro práci s touto knihovnou je potřeba využít knihovny session a nastavit v databázi tabulky pro průběžné ukládání dat.
Funkce:
insert - je metoda, která zjistí, zda se jedná o přidání jednoho produktu nebo více. Všechny produkty uloží. Pokud vše proběhne bez chyby, vrací TRUE, jinak FALSE.
_insert - konstruktor zkontroluje vstupní data a vytvoří pole dat s jedinečným id, který vrátí.
update - metoda slouží pro úpravu jednoho nebo více produktů. Pokud se data

upraví, uloží se celý košík. Pokud vše proběhne bez chyby, vrací TRUE, jinak FALSE.
_update - konstruktor zkontroluje data. Změny v datech upraví, v případě nastavení počtu na 0 odebere produkt. Pokud vše proběhne bez chyby, vrací TRUE, jinak FALSE.
_save_cart - konstruktor přepočte celkovou cenu a počet produktů. Uloží data do databáze. Pokud vše proběhne bez chyby vrací TRUE jinak FALSE.
total - metoda vrací celkovou cenu.
total_items - metoda vrací počet produktů.
contents - metoda vrací celý obsah košíku.
destroy - metoda nastaví celkový počet a cenu na nulu. Vymaže data z databáze.

Email[14] - je robustní knihovna pro práci s e-maily.
Funkce:
initialize - funkce inicializace slouží k nastavení vlastností e-mailu. Nastavení e-mail protokolu, povolení zalamování řádků, rozdělení e-mailu na části a další parametry.
clear - funkce slouží k vymazání všech údajů e-mailové zprávy i příloh, využívá se při cyklickém odesílání podobných zpráv.
from - funkce nastaví adresu a jméno odesílatele.
reply_to - funkce nastaví adresu a jméno, pro odpověď.
to - funkce nastaví adresu nebo adresy příjemců oddělené čárkami nebo jako pole.
cc - funkce nastaví adresu a jméno příjemce kopie.
bcc - funkce nastaví adresu a jméno příjemce skryté kopie.
subject - funkce nastaví předmět e-mailu.
message - funkce nastaví obsah e-mailu.
attach - funkce umožňuje připojit k e-mailu přílohu. Funkce připojí vždy jen jednu přílohu. Vstupním parametrem je cesta k příloze.
send - funkce slouží k odeslání e-mailu.

Encrypt[14] - je knihovna určená k šifrování dat. Data je možno zašifrovat pomocí klíče a následně dešifrovat. Využívá se funkce XOR. Další možností je hešování hesel pomocí funkce sha1 nebo md5, tyto metody jsou nevratné.
Funkce:
get_key - metoda upraví klíč na správnou velikost a vrací ho zašifrovaný přes md5.
set_key - metoda uloží klíč.
encode a decode - metody k zakódování a dekódování pomocí XOR.
mcrypt_encode a mcrypt_decode - metoda zašifruje data a přidá šum.
hash - funkce zašifruje vstupní data buď sha1 nebo md5.
sha1 - funkce zjistí, zda je vytvořena vlastní funkce sha1, pokud ne, tak zašifruje klasickou funkcí sha1.

Form_validation[14] - je knihovna ošetřující vstupní data proti vložení chybných formátů, spatných tvarů nebo velikosti. Zabrání uživateli vložit nesmyslné údaje.
Funkce:
set_rules funkce zkontroluje, zda vstupují data. Dále nastaví validační pravidla.
set_message funkce nastaví text chybového hlášení, umožňuje vložení vlastního chybového hlášení k daným datům.
set_error_delimiters - funkce nastaví formát chybového hlášení.
error - funkce vloží chybové hlášení do zvoleného formátu.
run - tato funkce zkontroluje data podle zadaných pravidel, pokud jsou správné, vrací TRUE, jinak FALSE.
Další funkce jsou funkce pro všechny validační pravidla.

FTP[14] - je knihovna umožňující přenos souborů na vzdálený server. Vzdálené soubory je možno přesouvat, přejmenovávat a mazat. Nepodporuje SFTP.
Funkce:
connect - funkce připojuje a přihlašuje k FTP serveru. Vstupním parametrem je pole s přihlašovacími údaji (hostname, uživatelské jméno, heslo, port). Návratovou hodnotou je TRUE nebo FALSE.
mkdir - funkce sloužící k vytvoření adresáře.
upload - funkce má čtyři vstupní parametry, a to adresu odkud soubor kopírovat, kam soubor kopírovat, formát kopírování a oprávnění souboru. Funkce zkontroluje připojení, zjistí zda soubor existuje, uloží soubor a nastaví oprávnění. Návratovou hodnotou je TRUE nebo FALSE.
download - vstupem funkce je adresa souboru, adresa kam se má uložit a formát kopírování. Návratovou hodnotou je TRUE nebo FALSE.
rename - funkce slouží k přejmenovávání souborů.
move - funkce provádí přesun souborů.
delete_file a delete_dir - funkce slouží k mazání souborů a adresářů.
list_files - vstupem do funkce je cesta zvoleného adresáře, návratovou hodnotou je seznam souborů.
chmod - funkce slouží k nastavení oprávnění.
mirror - funkce vytvoří obraz složky.
close - funkce ukončí připojení k serveru.

Image_lib[14] - je knihovna určená pro práci s obrázky. Umožňuje funkce na změnu velikosti, oříznutí obrázku a další funkce.
Funkce:
clear - metoda nastaví počáteční hodnoty třídy.

initialize - je metoda, do které vstupuje pole požadavků, s jakým obrázkem se bude pracovat, a co všechno se na něm bude měnit.
resize - funkce nastaví parametry pro změnu velikosti.
crop - funkce nastaví parametry pro oříznutí obrázku.
rotate - funkce nastaví parametry pro rotaci.
image_process_netpbm - funkce provede změnu velikosti, ořízne nebo otočí obrázek.
image_rotate_gd - funkce provádí rotaci.
image_mirror_gd - funkce provádí převrácení horizontální nebo vertikální.
watermark - metoda vrací buď obrázkový vodoznak nebo textový.
overlay_watermark - metoda vytvoří vodoznak z obrázku, u kterého lze nastavit krytí vodoznaku, barvu vodoznaku.
text_watermark - metoda vytváří textový vodoznak. U vodoznaku lze nastavit barvu, velikost textu, posunutí stínu a barvu stínu.
image_create_gd - funkce vytvoří obrázek po úpravách.
image_save_gd - funkce uloží obrázek.
image_display_gd - funkce vykreslí obrázek na obrazovku.
image_reproportion - funkce získá počáteční rozměry obrázku.
size_calculator - funkce vypočte chybějící rozměr nového obrázku tak, aby zůstal zachován stejný poměr. Návratovou hodnotou je pole všech rozměrů.
explode_name - funkce oddělí od sebe název a příponu obrázku.

Javascript[14] - CodeIgniter obsahuje knihovnu, která pomůže se společnými funkcemi, které budete chtít použít s Javascript. CodeIgniter nabízí snadné nastavování akcí, které se provádí za určitých událostí.
Funkce:
change, click, dblclick, error, mousedown, mouseout, keydown a keyup - jsou funkce, kterým lze přiřadit události ,které se následně vykonají.
hide a show - funkce nastavující způsob skrytí a zobrazení s možností nastavení rychlosti.
animate - funkce slouží k nastavení parametrů animací.

Pagination[14] - knihovna slouží k usnadnění práce se stránkováním. Je možné nastavit dynamické nebo pevné stránkování.
Funkce:
initialize - funkce nastaví parametry a formát stránkování.
create_links - metoda podle nastavení vytvoří příslušné odkazy pro listování mezi stránkami.

Parser[14] - je knihovna pro usnadnění práce s pseudoproměnnými, což je například

titulek, hlavička stránky nebo tělo stránky.
Funkce:
parse - metoda vytvoří pole, kde každé proměnné přiřadí určitý text. Ve wiew stačí vložit název proměnné mezi složené závorky. *parse_string* - metoda umí navíc pracovat s proměnnými, které jsou zastoupeny polem. Pokud má proměnná párový znak, zobrazí se data v poli postupně za sebou.

Session[14] - je knihovna pracující s relacemi. V relacích se ukládají informace o přihlášeném uživateli, jeho identifikace, kdy se přihlásil, odkud se přihlásil. Relace mohou být uloženy v cookie nebo v databázi, v podobě textu nebo zašifrované.
Funkce:
sess_create - vytvoří novou relaci.
sess_destroy - odstraní existující relaci.
userdata - funkce umožňuje načítat informace ze session.
setuserdata - funkce uloží informace do session.
all_userdata - funkce vyčte všechny data ze session.
unset_userdata - funkce odstraní část uložených dat.
set_flashdata - funkce pro uložení dat,data budou k dispozici pouze pro následující požadavek.
flashdata - funkce umožňuje načíst data.
keep_flashdata - funkce umožní data uschovat na delší dobu.

Typography[14] - je malá knihovna poskytující funkce pro formátování textu.
Funkce:
auto_typography - funkce formátuje text např. dvě pomlčky nahrazuje jednou dlouhou. Konce řádku převede na <br>.
format_characters - funkce je podobná funkci *auto_typography* slouží pouze k nahrazování znaků.

Unit_test[14] - je knihovna určená k testování. Knihovna zkontroluje formáty datových typů a výsledky daných proměnných. Nabízí funkce pro zobrazení výsledků.
Funkce:
set_test_items - funkce se využívá k nastavení parametrů testu (název testu, datový typ, předpokládaný typ, výsledek, název souboru, číslo řádku, poznámka)
run - funkce porovná buď dva doslovné názvy, nebo určí, zda zvolená data odpovídají požadovanému datovému typu.
report - funkce vygeneruje zprávu ze všech testů.
use_strict - funkce slouží k nastavení přísnosti kontroly. Při porovnání hodnoty bool 1 a TRUE bude výsledek pravdivý. V přísném režimu hodnota 1 nebude považována

stejně jako TRUE.
active - funkce spouští a vypíná testování.
result - funkce vrací neformátovaný výsledek testu.
set_template - funkce nastavuje formát zobrazení výsledku.

User Agent[14] - je knihovna obsahující funkce, které pomáhají zjistit informace o prohlížeči, mobilním zařízení, nebo když automat navštíví vaše stránky. Navíc můžete získat některé informace, jakož i jazyk a podporované znakové sady.
Funkce:
is_browser - funkce vrátí TRUE pokud se jedná o webový prohlížeč.
is_robot - funkce vrátí TRUE, pokud se jedná o mobilní zařízení.
is_mobile - funkce vrátí TRUE, pokud se jedná o robota.
is_referral - funkce vrátí TRUE, pokud jde o user agent z jiných stránek.
agent_string - funkce vrací řetězec obsahující všechny informace o uživateli.
platform - funkce vrací platformu prohlížeče webu.
browser, robot, mobile - funkce vrací název webového prohlížeče.
languages - funkce vrací používaný jazyk.

Xml[14] - knihovna sloužící ke komunikaci mezi dvěma počítači. Xml-rpc je část knihovny, která funguje jako klient. Xml-rpcs je část knihovny, která funguje jako server.
Funkce:
Xml-rpc
initialize - funkce slouží ke konfiguraci parametrů klienta.
server - funkce slouží k nastavení adresy a portu serveru.
timeout - funkce nastaví čas prodlevy, po které bude žádost zrušena.
method - funkce nastaví metodu, jakou chceme volat.
request - funkce vytvoří žádost ze zadaných údajů.
send_request - funkce odešle žádost na daný server se zadanými požadavky. Návratovou hodnotou je TRUE nebo FALSE.
display_error - funkce vrací chybové hlášení v případě selhání, zpráva je ve formátu řetězce.
send_error_message - funkce slouží k zaslání zprávy ze serveru na klienta. Prvním vráceným parametrem je číslo chyby a druhým je text chybové zprávy.
send_response - funkce umožňuje odeslat odpověď od serveru ke klientovi.
Xml-rpcs
initialize - funkce slouží ke konfiguraci parametrů serveru.
serve - metoda slouží ke spuštění serveru umožňující přístup k určitým metodám a funkcím.

Zip[14] - je knihovna určená pro práci se Zip soubory, umožňuje přidávat do Zip souborů, vytvářet a stahovat Zip soubory.
Funkce:
add_dir - funkce pro přidání adresáře.
add_data - funkce přidá data do souboru.
read_file - funkce si načte soubor a přidá ho do souboru Zip.
read_dir -funkce si načte adresář a přidá ho do souboru Zip. Adresářová struktura zůstane stejná.
get_zip - funkce slouží k získání dat ze Zip souboru.
archive - funkce uloží Zip soubor na zadané místo.
download - funkce stáhne soubor Zip.
clear_data - metoda vymaže data třídy.

2.1.2 Zend Framework

Alc[5] - poskytuje lehký a flexibilní seznam řízení přístupu ACL, pro realizaci oprávnění řízení. Obecně platí, že aplikace využívá seznamy ACL pro řízení přístupu k některým chráněným objektům.
Funkce:
addRole - metoda slouží k přidávání nových rolí, jedním z parametrů může být odkaz na roli od, která se bude dědit. Role může dědit od více rodičovských rolí. Pokud se oprávnění překrývají, role získá právo posledního přidaného rodiče.
getRole - metoda vrací, o jakou roli se jedná.
hasRole - metoda zjistí, zda je role uložena v registru. Návratovou hodnotou je TRUE nebo FALSE.
inheritsRole - vstupním parametrem této metody je role a možný dědic. Metoda zjistí zda role dědí od dědice. V případě dědění vrací TRUE.
removeRole - metoda slouží k odstranění role z registru.
removeRoleAll - metoda vymaže všechny role z registru.
addResource - metoda přidá nový zdroj. Jako druhý parametr může být zdroj, od kterého bude dědit.
add - metoda přidá nový zdroj, vstupem je název nového zdroje a druhou vstupní hodnotou je název zdroje, od kterého dědí.
get - metoda vrací buď identifikační číslo zdroje, nebo jeho název.
has - metoda vrátí TRUE, pokud zdroj existuje. Vstupem může být identifikační číslo zdroje nebo jeho název.
inherist - metoda slouží ke zjištění, zda určitý zdroj dědí od svých předků. Metoda

zkontroluje celý strom dědění.
remove - metoda odstraní zdroj a všechny jeho potomky.
removeAll - metoda odstraní všechny zdroje.
allow - metoda přidělí roli oprávnění přístupu do určitého zdroje. Zdrojem je controller, u kterého lze povolit pouze určité akce.
deny - metoda slouží k zakázání pravidel, má opačnou funkci jako allow.
removeAllow - metoda odstraní pravidlo povolující přístup dané roli k danému zdroji.
removeDeny - metoda odstraní pravidlo zakazující přístup dané roli k danému zdroji.
setRule - metoda prování operace požadované od *allow, deny, removeAllow, removeDeny. isAllowed* - metoda vrací TRUE v případě, kdy zadaná role má přístup k danému zdroji.
getRegisteredRole - metoda vrací všechny role.
getRole - metoda vrací všechny názvy parametrů rolí.
getResources - metoda vrací všechny názvy parametrů zdrojů.

Auth[5] - je autentizační knihovna určená k identifikaci, o jakého uživatele se jedná. Využívá se při registraci, přihlášení a odhlášení uživatele.
Funkce:
getInstance - metoda se využívá k načtení nového vzoru.
getStorage - metoda nastaví typ úložiště, jako základní je nastaveno session.
setStorage - metoda slouží k uložení informací o uživateli.
authenticate - metoda se využívá k ověření identity. Pokud je nějaká identita uložena, dojde ke smazání a uloží se nová.
hasIdentity - metoda vrací TRUE, pokud zadaná identita odpovídá identitě uložené v database.
getIdentity - metoda vrací identitu nebo null.
clearIdentity - metoda smaže uloženou identitu.

Application[5] - knihovna slouží pro načtení a správu základních prvků aplikace bootstraping, zařízení pro opakování zdrojů. Stará se o nastavení PHP prostředí a zavádí automatické načítání v základního nastavení.
Funkce:
getEnvironment - metoda vrací prostředí aplikace.
getAutoloader - metoda načte počáteční nastavení.
setOptions - metoda načte konfigurační nastavení. Nastaví php, patch, jmenný prostor pro autoloader a bootstrap.
getOptions - metoda načte nastavení aplikace.
hasOption - metoda zkontroluje, zda je základní nastavení načteno.
getOption - metoda vrací jediné nastavení.

mergeOptions - metoda sloučí dvoje nastavení.
setPhpSettings - metoda provede nastavení php.
setAutoloaderNamespaces - metoda nastaví jmenné prostory.
setBootstrap - metoda nastaví soubor bootstrapt podle zadané cesty.
getBootstrap - metoda načte soubor bootstrapt.
bootstrap - metoda načte prvky ze souboru bootstrapt.
run - metoda spustí soubor bootstrapt.

Barcode[5] - knihovna slouží ke generování čárkových kódů. Skládá se ze dvou částí, z objektů čárových kódů a z vykreslovačů.
Funkce:
factory - metoda slouží k vytvoření zdroje pro vytvoření čárkového kódu. Vstupem může být pole nebo čtyři vstupní parametry. V nabídce je mnoho variant čárových kódů. Jedním vstupním parametrem je text zobrazovaný pod čárkovým kódem.
makeBarcode - metoda vytvoří instanci čárového kódu.
render - metoda vyrendruje obrázek čárového kódu.
draw - metoda vykreslí čárový kód.

Config[5] - je navržen tak, aby usnadnil přístup a používání konfiguračních dat v rámci aplikace. Poskytuje vnořené vlastnosti objektu uživatelského rozhraní pro přístup k těmto konfiguračním datům v kódu aplikace. Konfigurační data mohou pocházet z různých sdělovacích prostředků na podporu hierarchického ukládání dat. V současné době Zend_Config nabízí adaptéry pro konfigurační údaje, které jsou uloženy v textových souborech s Zend_Config_Ini a Zend_Config_Xml.
Funkce:
get - funkce načte zvolenou hodnotu. Pokud hodnota není nastavena vrací základní nastavení.
toArray - metoda vrací pole s parametry nastavení.
getSectionName - metoda vrací názvy připojených section.
merge - funkce slouží ke sloučení dvou konfiguračních souborů. Stejné položky nahradí údaji z nového souboru.
setReadOnly - metoda slouží k nastavení konfiguračního souboru do stavu jen pro čtení, bude zamezeno novému zapisování a slučování.

Date[5] - nabízí podrobný, ale jednoduchý API pro práci s daty a časy. Jeho metody přijímají širokou škálu typů, včetně dílčích částí, v mnoha kombinacích nabízí mnoho funkcí a možností nad rámec stávajících PHP funkcí. Zend_date podporuje zkrácené názvy měsíců v mnoha jazycích.
Funkce:

get - funkce vrací datum a čas v mnoha různých formách. Měsíce číslem, zkráceným nebo celým názvem v mnoha jazycích.
set - funkce nastaví nové datum jako nové datum nebo pouze jako část data.
add - funkce přičte k danému datu určitou časovou hodnotu(den, hodinu, rok).
sub - funkce odečte od daného data zadaný čas, funkce je inverzní k funkci *add*.
copyPart - funkce vytvoří kopii zadaného data.
compare - funkce porovná datum se stávajícím datem. Vrátí -1 pokud datum je dřívější, 0 pokud jsou data stejná a 1 pokud datum teprve bude.
equals - je funkce, která umožňuje porovnat části dvou dat. Při porovnání dnů v měsíci jsou čísla stejná, funkce vrací TRUE, ale při porovnání stejných dat na shodu roků se liší, a funkce vrací FALSE.
isEarlier a isLater - funkce vracejí TRUE nebo FALSE podle splnění podmínky porovnání starší nebo novější. U této funkce lze také porovnávat dílčí části dat.
isToday, isTomorrow a isYesterday - funkce zkontroluje, zda je dané datum dnešní, zítřejší nebo včerejší.

Filter[5] - poskytuje sadu běžných datových filtrů. Poskytuje jednoduchý filtrační mechanizmus, zřetězení, které může použít více filtrů v definovaném pořadí.
Funkce:
addFilter - metoda přidá filtr do řetězce.
appendFilter - metoda přidá nový filtr na konec řetězce.
prependFilter - metoda přidá nový filtr na začátek řetězce.
getFilters - metoda načte všechny filtry.
filter - metoda vrátí filtrovanou hodnotu přes všechny filtry v řetězci. Filtry se spouští v pořadí, v jakém byly přidány.
getDefaultNamespaces - metoda vrací nastavení jmenného prostoru.
setDefaultNamespaces - metoda nastaví jmenný prostor.
addDefaultNamespaces - metoda přidá nový jmenný prostor.
hasDefaultNamespaces - metoda vrací TRUE, pokud je nastaven jmenný prostor.
filterStatic - metoda odfiltruje data. Na data může být aplikován řetězec filtrů.

Form[5] - je rozsáhlá knihovna, která zjednodušuje vytváření formulářů a manipulaci s nimi ve webové aplikaci. Plní úkoly, vstupní filtraci, ověřování, uspořádání a vykreslení formuláře.
Funkce:
setConfig - funkce nastaví formulář podle parametrů v konfiguračním souboru.
addAttribs a setAttrib - funkce přidá a nastaví základní data.
setAction a getAction - funkce nastaví akci a vrací řetězec akcí.
addSubForm, setSubForms a getSubForm - funkce pracují s částí formuláře.

addErrorMessage a setErrorMessage - funkce vytváří a přidává chybové hlášení.
addDecorator, setDecorator, getDecorator, removeDecorator a clearDecorators - funkce slouží pro práci s nastavením formátování výsledného zobrazení.
render - metoda vezme všechny dekorátory a vygeneruje z nich výsledné zobrazení.
valid - metoda zjišťuje, zda prvek nebo část formuláře je správná.

Layout[5] - umožňuje obalit celý obsah aplikace do layoutu, který obvykle představuje zobrazovací šablona. Implementace layoutu ve frameworku se skládá z několika tříd, které jsou součástí balíčku Zend_Layout. Jedná se zejména o samotnou třídu Zend_Layout a třídy, které napomáhají integraci layoutu do MVC implementace, která je v Zend Frameworku.
Funkce:
startMvc - metoda sloužící k inicializování s využitím MVC.
setOptions a setConfig - knihovny využívající se k nastavení layoutu.
setLayout a getLayout - funkce nastavující a vracející název layoutu.
setLayoutPath a getLayoutPath - metoda nastaví cestu k šabloně s rozložením zobrazení.
disableLayout a enableLayout - *setView* - metoda nastaví, které view se bude zobrazovat.
render - metoda nastaví cestu k šabloně a předá data do view k vykreslení.

Log[5] - je knihovna, a slouží k zaznamenávání událostí. Knihovna nabízí ukládání do souborů i do databáze. Knihovna odesílá zprávy a umožňuje jejich filtrování. Událost, kterou chceme zaznamenat, dostane přidělený stupeň priority.
Funkce:
factory - funkce nastaví zaznamenávání v závislosti na nastavení, do něhož může zapisovat jeden nebo více zapisovatelů.
log - funkce předává zprávu se zadanou prioritou.
addPriority - funkce slouží k přidávání nových priorit. Zadává se název a hodnota priority. V základu je 8 priorit předdefinováno. Stávající priority nelze přepisovat.
addFilter - funkce slouží k filtrování zpráv podle zadané priority.
addWriter - funkce slouží k přidání nového zapisovatele. Zapisovatel se stará o přijetí zprávy a samotné zapsání do souboru nebo do databáze.
setEventItem - funkce slouží k nastavení nové hodnoty, která se přidá ke každé nově zaznamenané zprávě. Zadává se název položky a hodnota.

Gdata[5] - je knihovna spolupracující s knihovnami Google a poskytuje zobrazení řady aplikací nabízených googlem. Knihovna umožňuje načítat, posílat, aktualizovat a mazat data pomocí standardního protokolu http. Knihovna Gdata obsahuje

řadu dílčích knihoven:

- Zend_Gdata_Calendar - google kalendář
- Zend_Gdata_Spreadsheets - správa tabulky sdílené s více uživateli
- Zend_Gdata_Docs - správa prezentací, tabulek nebo textových dokumentů
- Zend_Gdata_YouTube - knihovna pro přístup na YouTube s možností přehrávání
- Zend_Gdata_Photos - knihovna pracující s daty programu Picasa
- a další

Funkce:

Kalendář

getCalendarEventFeed - funkce načte události z kalendáře.
getCalendarEventEntry - funkce po zadání ID načte jednu událost z kalendáře.
getCalendarListFeed - funkce vrátí seznam kalendářů.
insertEvent - funkce slouží k vložení události do kalendáře.

Docs

getDocumentListFeed - funkce vrátí seznam dokumentů. Podle adresy vrací tabulku nebo jiné dokumenty.
uploadFile - funkce slouží k nahrání dokumentů a převedení do dokumentu google. Jako název se použije název dokumentu.

Picasa

getUserFeed - funkce načte albumy, fotky a popisů spojených s daným uživatelem.
getAlbumFeed - funkce načítá alba od zadaného uživatele.
getPhotoFeed - funkce načte fotografie a popisky ze zvoleného alba.
getAlbumEntry - funkce načte album podle zadaného ID.
getPhotoEntry - funkce načte fotografii podle zadaného ID.
getCommentEntry - funkce načte komentář.
insertAlbumEntry - funkce slouží k vytvoření nového alba.
insertPhotoEntry - funkce slouží k vložení nového fotografie.
insertCommentEntry - funkce vloží nový komentář k fotografii.
deleteAlbumEntry, deletePhotoEntry a deleteCommentEntry - funkce k odstranění alba, fotografie a komentáře.

Memory[5] - je malá knihovna určena pro správu dat v prostředí s omezenou pamětí. Paměťové objekty jsou generovány správcem paměti, na vyžádání je můžeme vyměnit nebo načíst, když je to nutné.
Funkce:
factory - funkce vytvoří odkaz na soubor, do kterého se budou data ukládat.

Mail[5] - poskytuje zobecněné funkce vytvoření a odeslání textové i MIME vyhovující vícedílné e-mailové zprávy. Pomocí této knihovny lze maily i přijímat.

Funkce:
setBodyText a getBodyText - metody umožňují nastavit a zpětně získat text e-mailu ve formátu textu.
setBodyHtml a getBodyHtml - metody umožňují nastavit a zpětně získat text e-mailu, který je v html formátu.
addAttachment - metoda připojí přílohu k e-mailu.
addTo - metoda přidává k e-mailu příjemce jako pole nebo řetězec.
addCc - metoda přidává adresu adresáta pro přijetí kopie e-mailu.
addBcc - metoda umožňuje odeslat skrytou kopii.
getRecipients - metoda vrací seznam všech přidaných příjemců.
setFrom - metoda nastaví adresu odesilatele.
getFrom - metoda vrací nastaveného odesilatele.
clearFrom - metoda vymaže odesilatele.
setSubject - metoda nastaví předmět emailu.
clearSubject - metoda vymaže předmět emailu.
addHeader - metoda umožňuje přidat hlavičku e-mailu.
send - metoda slouží k samotnému odeslání e-mailu.

Pdf[5] - pomocí této knihovny je možné načíst, vytvářet, upravovat a ukládat dokumenty. Takto může pomoci, každý PHP aplikace dynamicky vytvářet PDF dokumenty úpravou stávajících dokumentů nebo vytváření.
Funkce:
parse - vytvoří soubor pdf podle vloženého řetězce parametrů.
load - funkce načte již existující pdf k dalšímu zpracování.
save - funkce slouží k uložení nového souboru pdf, nebo je možno k již existujícímu souboru přidat data na konec.
rollback - funkce vrátí počet revizí provedený v pdf souboru.
DrawLine - funkce slouží k nakreslení čáry, zadávají se krajní body.
drawRectangle - funkce slouží k vykreslení obdélníku.
drawRoundedRectangle - funkce slouží k vykreslení obdélníku se zaoblenými rohy.
DrawText - funkce vykreslí text do řádku ze zadaných počátečních souřadnic.
drawImage - funkce vykreslí do souboru obrázek.

Paginator[5] - je knihovna sloužící ke stránkování dat, umožňuje načítat pouze část dat. Zpřehledňuje a zrychluje zobrazení.
Funkce:
factory - funkce vytvoří objekt stránkování.
count - funkce vrací celkový počet stránek.
getpages - metoda vrací stránku ze zvoleného rozsahu.

setCurrentPageNumber - metoda nastaví číslo aktuální stránky, v základu je nastavena první stránka.
setItemCountPerPage - metoda nastaví počet prvků na stránku.
setView - metoda nastaví hodnoty pro vykreslení.

Validate[5] - je knihovna starající se o kontrolu vložení správných dat. V případě vložení špatných dat vrací validátory FALSE a zprávu vysvětlující co bylo zadáno špatně.
Funkce:
addValidator - funkce přidá do řetězce nový validátor s nastavenými parametry.
isValid -funkce zkontroluje data podle zadaných validátorů. Výsledkem je TRUE nebo FALSE.
getMessages - funkce vrací zprávy komentující špatné zadání dat.
setMessage - funkce nabízí možnost vlastního nastavení celé chybové zprávy.
is - metoda, která se využívá k snadnému a jednoduchému ověření. Zadává se pouze data pro zkontrolování a validátor, pomocí kterého se provede kontrola.
setMessageLength a getMessageLength - metoda nastaví a vrací omezení délky chybové zprávy. V případě zkrácení zprávy je konec nahrazen „. . .“.

Session[5] - je knihovna nabízející práci s údaji relací. Tyto relace doplňují cookie, liší se však tím, že data jsou uloženy na straně serveru a ne u klienta. Sesion se využívá při uchování informací o přihlášených uživatelích.
Funkce:
setOptions - *sessionExists* - funkce zjistí, zda zadaná relace existuje.
start - metoda spustí relaci. *isStarted* - funkce zjisti zda je zvolená relace spuštěna.
stop - metoda zakáže zápis do relace.
setId a getId - funkce nastaví a vrací ID relace.
writeClose - funkce vypne relaci.
destroy - funkce slouží ke zrušení dat relace.
getIterato - metoda vrací pole názvů relací.

2.2 Volně dostupné knihovny

2.2.1 CodeIgniter

Tank auth library[19]
Jedná se o autentizační knihovnu, k zakódování hesel využívá knihovnu phpass, umožňuje jednorázovou registraci, nebo bezpečnější aktivaci s ověřením e-mailem.

Při opakovaném špatném přihlášení nabízí CAPCHA. Možnost nastavení zapamatování přihlášeného. Má funkce pro změnu hesla, změnu ,mailu, udělení BAN uživateli.
konyukhov.com/soft/tank_auth

Ajax pagination library
Knihovna slouží k vytvoření stránkování Ajax s použitím JQuery.
github.com/neotohin/CodeIgniter-Ajax-pagination-Library

Captcha library
Knihovna slouží k vytvoření capcha códu, u kterého umožňuje nastavi písmo, hustotu zašumění.
github.com/neotohin/Captcha-library-for-CodeIgniter

Bonfire library
Tato knihovna je rozsáhlá šablona s mnoha základními funkcemi, jako je emailový systém, vývojové prostředí, obsah, statistiku a autentizaci.
github.com/neotohin/Bonfire

Facebook Ignited library
Knihovna umožňuje propojení s Facebook.
github.com/DarkProspectGames/Facebook-Ignited

Generic Table editor
Je knihovna umožňující snadný způsob pro editaci a úpravu databázových tabulek.
www.bird.li/TableEditor

2.2.2 Zend framework

Zend framework Facebook library 1.0.0
Je knihovna sloužící k připojení a provázání aplikace s Facebook.
www.phpkode.com/scripts/item/zend-framework-facebook-library/

GDataGmailer 1.0
Tato knihovna slouží ke stažení kontaktů z Gmail.
www.phpkode.com/scripts/item/gdatagmailer

OAuth Twitter 0.3
Tato knihovna nabízí řadu funkcí k propojení a práci s Twitter.

www.phpkode.com/scripts/item/oauth-twitter

2.3 Využití knihoven při návrhu aplikací

Tab. 2.1: Využití knihoven při návrhu vzorových aplikací

Náležitosti	**Knihovny**	
	CodeIgniter	**Zend**
Správa uživatelů	Tank_auth* Session Database Template	Auth Acl Registry Db Layout
Přidělování rolí	Database Template	Db Layout Form
Vyhledávání	Database Template	Db Layout Form
Filtrování	Validation	Filter
Ošetření vstupních dat	Validation	Validate Filter
Stránkování	Pagination Database	Paginator Db
Email	Email Database	Mail Db

* Tank_auth - není základní knihovna CodeIgniter.

3 POROVNÁNÍ FRAMEWORKŮ ZEND A CODEIGNITER

Pro porovnáni frameworku jsem si vybral dva frameworky, a to Zend framework 1.12 a CodeIgniter 2.1.3.

Obr. 3.1: Logo CodeIgniter a Zend framework[5][14]

3.1 Vlastnosti

Automatické generování kódu

Generování kódu by byla velmi užitečná věc a usnadnilo by to spoustu práce, protože mnoho kódu se v programech opakuje a liší se jen malými změnami. Automatické generování kódu by ušetřilo dost času a zamezilo mnoha chybám. CodeIgniter žádnou takovou funkci bohužel nenabízí. Zend framework nabízí knihovnu Tool, která pomocí příkazové řádky vygeneruje základní adresářová uspořádání. Pomocí této knihovny do vytvořeného adresáře lze vygenerovat prázdné soubory controleru, akci, formulář, view a model. Tyto funkce jsou zajímavé, ale ne často využívané.

PHP

Jak se postupně vyvíjejí nové verze frameworků, mění se i požadavky na novější verze PHP. U CodeIdniteru se vyžaduje verzi PHP 5.1.6 a novější. Zend vyžaduje verzi PHP 5.2.4 a novější.

Uživatelské oprávnění

U tohoto požadavku jsem narazil na nedostatek CodeIgniter. Nenabízí žádnou knihovnu pro práci s uživateli a jejich oprávněními. Zend framework nabízí knihovnu

Zend_Acl, která umožňuje správu rolí. Knihovna nabízí možnost dědičnosti vlastností od jedné nebo více rolí. Knihovna taktéž nabízí možnost nastavení oprávnění vytvořit, číst, aktualizovat a smazat. Dle potřeby je možno zvoleným uživatelům udělovat výjimky v oprávnění pro určité funkce.

Validace

Je funkce sloužící k určení, zda vložená data jsou ve formátu, jaký jsme požadovali. Validace obou frameworků umožňují kontrolu, zda se jedná o číslo, datum, řetězec složený pouze z písmen, e-mail nebo IP adresu. Validace kontroluje délku řetězce. Zamezí vložení nekorektních znaků. Umožní kontrolu při registraci, zda zadané hesla jsou schodná. CodeIgniter nabízí všechny základní pravidla pro validaci. U kontroly prku lze použít více pravidel. Zend má ve své knihovně řadu pravidel, které CodeIgniter nenabízí, jako je kontrola, zda se jedná o číslo kreditní karty, poštovní směrovací číslo nebo o číslo určitého čárového kódu.

Email

Oba porovnávané frameworky mají knihovnu pro odesílání emailů. Umožňují posílat maily s přílohami. Zend navíc nabízí možnost čtení mailů z místních nebo vzdálených poštovních schránek a umožňuje následné zpracování.

Datum a čas

Porovnávané frameworky umožňují práci s časy a nabízí různé formáty pro samotné zobrazení. CodeIgniter pro práci s časovými údaji nabízí helper. CodeIgniter navíc oproti Zend nabízí knihovnu pro tvorbu kalendáře.

Jazyk

Vybrané frameworky nabízí možnost nastavení a zobrazování v různých jazycích. CodeIgniter umožňuje ukládat jazykové formáty v souboru (.php). Zend umožňuje pracovat s více jazykovými formáty(.csv .ini .Tbx .Xml).

PDF

Zend má knihovnu pro práci se soubory PDF. Může soubory číst, upravovat i tvořit od začátku. Knihovna umožňuje měnit pořadí stránek. Má funkce pro kreslení jednoduchých tvarů a vkládání obrázků. CodeIgniter knihovnu pro práci s PDF nenabízí.

Fotky

CodeIgniter má knihovnu pro práci s obrázky. Funkce umožňují měnit velikost obrázku, vytvořit náhledový obrázek, oříznout, otočit obrázek a vložit vodoznak do obrázku. Zend framework takovou funkci nenabízí.

Stránkování

Je knihovna, kterou mají oba frameworky. Stránkování slouží k zpřehlednění a zrychlení načítání stránek. Z požadavku na zobrazení na stránce se zobrazí jen část a na další prvky se dostaneme pomocí čísla následující stránky nebo pomocí tlačítka další.

Nahrávání souborů

Oba frameworky nabízí knihovny pro vkládání souborů pomocí protokolu HTTP a FTP. Při nahrávání souboru dochází ke kontrole, zda soubor odpovídá předpokladům toho, co je povoleno. Vkládání lze omezit u obrázků rozměry fotky, dále podle velikosti a typu souboru. Podle nastavení se uloží s původním názvem, nebo bude přejmenován.

ZIP

CodeIgniter nabízí knihovnu pro práci s archivy ZIP. Umožňuje vytváření a přidávání dat do ZIP souborů. Zend knihovnu pro práci s archivy nenabízí.

Měna

Zend nabízí knihovnu pro práci s měnou, která slouží k početním operacím. Umožňuje přepočty měn. Nabízí řadu šablon pro formátování měn.

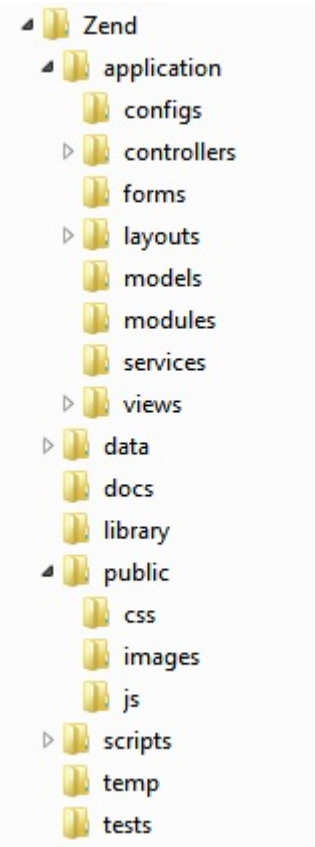
3.2 Dokumentace

Oba frameworky nabízí dostatečnou dokumentaci a návody k tvorbě vlastní aplikace. **Zend** má na svých oficiálních stránkách framework.zend.com příručku pro začínající programátory, kde ve stručnosti vysvětluje MVC, jak vytvořit první projekt, formulář a návod na databázovou tabulku. Zend má přehlednou dokumentaci všech knihoven a jejich funkcí, knihovny jsou popsány pro více verzí pro ZF1 a pro novější ZF2. Dále nabízí uživatelskou příručku s řadou řešených příkladů. Zend nabízí na svých stránkách možnost školení, které ovšem není v České republice. Na internetu je možno najít mnoho fór, kde je vyřešena spousta problémů, a

které umožňují vznést dotaz k problému. **CodeIgniter** uživateli na svých stránkách ellislab.com/codeigniter nabídne taktéž uživatelskou příručku s přehledným zobrazením knihoven. Součástí uživatelské příručky je návod na instalaci a návod, jak aktualizovat svoji předchozí verzi na nejnovější. Jako vhodný považuji úvodní seznámení se samotným programem a MVC. Krátký video tutoriál s počátečními kroky a základním seznámením s aplikací. Má přímo na svých stránkách fórum.

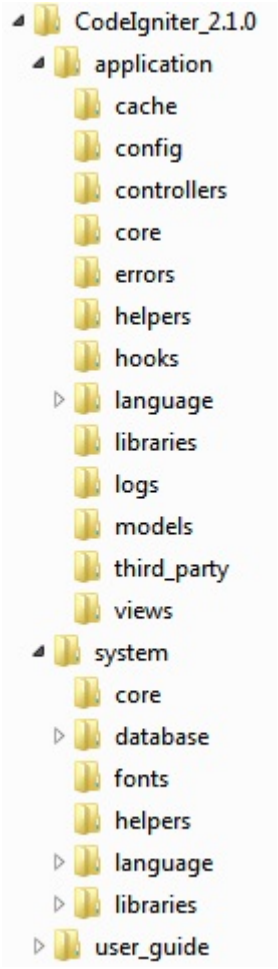
3.3 Adresářová struktura

Struktury ukládání souborů aplikace jsou odlišné. U **Zend** se hlavní část aplikace ukládá do složky *application*, vlastní knihovny se vkládají mezi knihovny Zend. Do složky *public* se ukládají všechny obrázky, css a javascript soubory.



Obr. 3.2: Adresářová struktura - Zend Framework

CodeIgniter má celou aplikaci ve složce *application*, do které vkládáme vše, co se týká naší aplikace, a do složky *system* není nutné zasahovat. Vlastní vytvořené knihovny a helpery se ukládají do složek v *application* a nemíchají se s knihovnami CodeIgniter, při změně jsou nadřazeny knihovnám původním. Obrázky a css soubory ukládáme do složky *assets*.



Obr. 3.3: Adresářová struktura - CodeIgniter

3.4 Vývoj

Na vývoji obou frameworků se stále pracuje, vytváří se aktualizace, frameworky se rozšiřují o nově potřebné funkce a odstraňují se chyby. **Zend** má aktuálně verzi Zend 1.12.3 a verzi Zend 2.1.5 . **CodeIgniter** přišel s novou verzí CodeIgniter 2.1.0 v listopadu 2011, aktuální verzí je CodeIgniter 2.1.3.

3.5 Srovnávací tabulka

Tab. 3.1: Porovnání Zend framework a CodeIgniter

FrameWork	**Zend**	**CodeIgniter**
PHP (min)	5.24	5.16
MVC	ANO	ANO
Verze	1.12.3	2.1.3.
RSS	ANO	ANO
Velikost (přibližně)	55MB	4MB
Knihy	ANO (CZ)	ANO (EN)
Mail	ANO	ANO
PDF	ANO	NE
Uživatelské oprávnění	ANO	NE
Databáze	MySQL, MSSQL, Oracle, PostgreSQL, IMB DB2	MySQL, MySQLi, MS SQL, PostgreSQL, Oracle, SQ-Lite, ODBC
Licence	zdarma	zdarma

3.6 Zhodnocení vlastností

Vybrané frameworky obsahují základní knihovny pro práci s funkcemi, které se využijí na každých stránkách. Velmi užitečné jsou knihovny pro práci s emaily, jako další se mi líbí knihovny pro práci s jazyky, protože již většina stránek využívá přepínání mezi zobrazením jendnotlivých jazykových mutací stránek. Jako výhodu CodeIgniter považuji knihovnu pro práci s fotkami, dle mého názoru tato knihovna nemá moc velké uplatnění u většiny stránek. Oproti Zend nabízí knihovnu pro práci se soubory ZIP. Zend framework nabízí ve srovnání s CodeIgiter mnohem víc užitečných knihoven. Nejužitečnější knihovnou, kterou nabízí Zend navíc, je knihovna pro práci s uživatelskými oprávněními, která se využije ve všech stránkách. Jako často používanou bych zařadil knihovnu pro práci a úpravu PDF souborů. Zend má knihovnu s funkcemi pro práci s měnami, která lze využít u internetových obchodů. Porovnávané frameworky mají výbornou a přehlednou dokumentaci a stručný návod jak začít. U Zend jako velmi užitečnou považuji knihu *Zend Framework*, která

je dostupná na českém trhu. Při seznamování s CodeIgniter mi pomohl slovenský video tutoriál. U CodeIgniter považuji za výhodné jeho uložení vlastní aplikace do samostatného adresáře mimo systémové soubory. Oba frameworky jsou neustále vylepšovány a aktualizovány, jejich rozdílná velikost dle mého názoru nehraje příliš velkou roli. Pro práci bych si vybral Zend framework pro jeho větší počet nabízených knihoven, které jsou dle mého názoru užitečnější.

4 VZOROVÉ APLIKACE

4.1 Využité programy

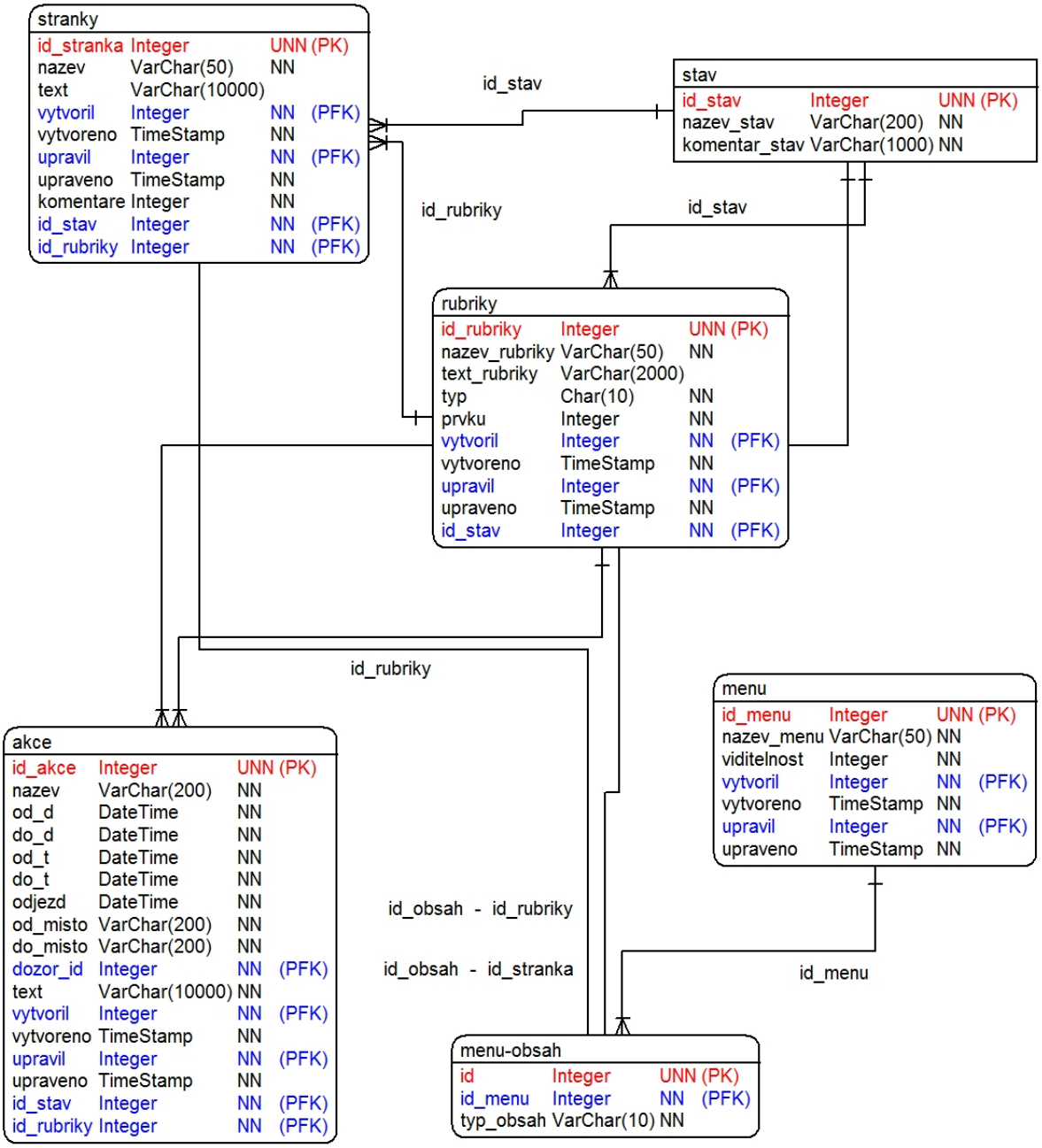
XAMPP[17] - Je volně šiřitelný program obsahují řadu funkcí pro zprovoznění domácího serveru. Instalace a práce v tomto programu je jednoduchá. Vytvoření vlastní databáze v phpMyAdmin je snadné a rychlé, prostředí je možno nastavit do českého jazyka. XAMPP obsahuje:

- Apache
- Mysql
- phpMyAdmin
- ProFTPD
- Mercury
- OpenSSl

PSPad[18] - Je volně šiřitelný univerzální editor. Umožňuje ukládat soubory v široké škále formátů a kódování. PSPad barevně zvýrazňuje syntaxe. Nabízí vyhledávání s možností nahrazení. Obsahuje editor TopStyle Lite pro editaci CSS. Umožňuje současně práci ve více dokumentech. PSPad nabízí řadu balíčků na rozšíření svých možností.

4.2 Databáze

V prostředí phpMyAdmin jsem vytvořil jednu databázi pracující s oběma frameworky. Většinu tabulek v databázi využívají oba frameworky. CodeIgniter má v databázi přidány vlastní tabulky, které využívá k přihlašování uživatelů. Ci_sessions, user_autologin jsou tabulky, které jsou součástí autentizační knihovny pro CodeIgniter. Dále tabulku práva a zdroje využívá pro svá oprávnění také CodeIgniter, u Zend jsou oprávnění řešena skrz Acl.php, orprávnění přístupu je vysvětleno v kapitole 4.3.6. Soupis a využití databázových tabulek je vidět v tabulce4.1. Na obrázku4.1 je ER diagram zobrazující entity týkající se obsahu aplikace. Další diagramy jsem zařadil do příloh.



Obr. 4.1: ER digram obsahu aplikací

Tab. 4.1: Využití databázových tabulek

Tabulka	**Zend**	**CodeIgniter**
akce	ANO	ANO
ci_sessions	NE	ANO
menu	ANO	ANO
menu-obsah	ANO	ANO
prava	NE	ANO
role	ANO	ANO
rubriky	ANO	ANO
stav	ANO	ANO
stranky	ANO	ANO
users	ANO	ANO
user_autologin	NE	ANO
user_profiles	ANO	ANO
zdroj	NE	ANO

4.3 Zend

Po instalaci frameworku Zend a vytvoření adresářové struktury pro přehledné ukládání částí kódu samotné aplikace, můžeme začít se samotnou prací na aplikaci. Vlastní program se nachází ve složce *application* adresářová struktura viz. obrázek3.2. V této struktuře jsou soubory roztříděny podle typu na *models*, *controllers* a *views*.

4.3.1 Konfigurace

Konfiguraci jsem provedl pomocí souboru application.ini nacházející se ve složce *configs*. Kód konfiguračního souboru je vidět na obrázku 4.2. V tomto souboru jsem nastavil přístup do databázové tabulky. Přidal jsem odkazy, odkud se budou načítat formuláře a základní šablona layout.

```
[production]
phpSettings.display_startup_errors = 0
phpSettings.display_errors = 0
includePaths.library = APPLICATION_PATH "/../library"
includePaths.forms = APPLICATION_PATH "/../forms"
bootstrap.path = APPLICATION_PATH "/Bootstrap.php"
bootstrap.class = "Bootstrap"
appnamespace = "Application"
resources.frontController.controllerDirectory = APPLICATION_PATH "/controllers"
resources.frontController.params.displayExceptions = 0

resources.db.adapter = "Pdo_MySQL"
resources.db.params.host = "localhost"
resources.db.params.username ="root"
resources.db.params.password = ""
resources.db.params.dbname =  "skautcz"
resources.db.params.charset = "utf8"
resources.db.isDefaultTableAdapter = true

resources.layout.layout = "layout"
resources.layout.layoutPath = APPLICATION_PATH "/layouts/scripts/"
resources.view.doctype = "XHTML1_STRICT"
[staging : production]

[testing : production]
phpSettings.display_startup_errors = 1
phpSettings.display_errors = 1

[development : production]
phpSettings.display_startup_errors = 1
phpSettings.display_errors = 1
resources.frontController.params.displayExceptions = 1
```

Obr. 4.2: Zend konfigurace

4.3.2 Filtrování a ošetření vstupních dat

K ošetření vstupních dat se u Zend nabídly dvě knihovny, a to *Filter* a *Validate*. Tyto knihovny se nemuseli nijak incializovat. Ošetření vstupních dat se nastavuje přímo ve formuláři pro každou proměnou zvlášť. K proměnné se připojují pravidla validace a zvolené filtry. Samotná kontrola se provádí v controlleru. Přepsání chybového hlášení lze přímo při zadávání validátoru pro každý políčko zvlášť, nebo přímo do knihovny s validátory. Na obrázku 4.3 je ukázáno přidání validátorů a filtrů do souboru *Register* ve složce *forms*.

- required - Vyžaduje naplnění proměnné daty.
- Digits - Hodnota musí být číslo.
- EmailAddress - Validátor kontroluje správnost vložení emailové adresy.
- StringLength - Určuje počet písmen. První parametr určuje, zda se má zobrazovat chybová hláška. Druhý parametr určuje rozsah nebo přesný rozměr.
- filtr Alpha - Odfiltruje vše kromě písmen.

- filtr StripTrim - Odstraní bílé znaky na začátku a na konci.
- filtr StripTags - Odstraní html a xml tagy.

```
$jmeno = new Zend_Form_Element_Text('jmeno');
$jmeno->setLabel('Jméno :')
->setRequired(true)->addFilter('StripTags')->addFilter('StringTrim')
->addFilter('Alpha')->addValidator('NotEmpty',true,array('messages'
=>array('isEmpty' => 'Jméno musí být vyplněno!')));
$prijmeni = new Zend_Form_Element_Text('prijmeni');
$prijmeni->setLabel('Příjmení :')
->setRequired(true)->addFilter('StripTags')->addFilter('StringTrim')
->addFilter('Alpha')->addValidator('NotEmpty',true,array('messages'
=>array('isEmpty' => 'Příjmení musí být vyplněno!')));
$prezdivka = new Zend_Form_Element_Text('prezdivka');
$prezdivka->setLabel('Přezdívka :')
->addFilter('StripTags')->addFilter('StringTrim')->addFilter('Alpha');
$mobil = new Zend_Form_Element_Text('mobil');
$mobil->setLabel('Mobilní číslo :')
->addValidator('NotEmpty',true,array('messages'
=>array('isEmpty' => 'Mobilní číslo musí být vyplněno!')))
->setRequired(true)->addFilter('StripTags')->addFilter('StringTrim')
->addValidator('Digits');
```

Obr. 4.3: Zend Validace dat uživatele

4.3.3 Stránkování

Zend nabízí knihovnu *Paginator* pro tvorbu stránkování. K samotnému přidání stránkování bylo nutné upravit controller a view. Dalším krokem bylo vytvořit si view, obsahující navigační lištu umožňující listování mezi stránkami. V části kódu z controlleru *IndexController*, který je na obrázku 4.4, je nutné načíst všechny publikované stránky, vytvořit objekt stránkování, nastavit počáteční zobrazovanou stránku a počet zobrazovaných prvků na stránce. V zobrazení, ve kterém stránkování budeme provádět jsem nakonec vložil view s navidací. V samotném view *strankovani* jsou vytvořeny odkazy šipek a číselné odkazy na stránky. Podle aktuální pozice je zakázáno odkazování na aktuální stránku a v případě první nebo poslední stránky nefungují příslušné odkazy šipek.

4.3.4 Vyhledávání

Vyhledávání je využito k vyhledávání uživatelů. Přístup do vyhledávání je pomocí odkazu v horním menu, oprávnění k vyhledávání uživatelů mají role *vedoucí*, *hl.vedoucí* a *admin*. Stránka vyhledávání se skládá ze dvou částí, v horní části je formulář pro zadání požadavku, v dolní polovině obrazovky je tabulka s výsledky.

```
$data = new Application_Model_Stranky();
$datastranky = $data->publikovanyStranka();
$page=(int) $this->getrequest()->getParam('page',1);  //uvodní stranka
$stranky = Zend_Paginator::factory($datastranky);    // vytvoření objektu
$stranky->setCurrentPageNumber($page); //nastaveni první stranky
$stranky->setItemCountPerPage(3);  // pocet prvku na stranku
$this->view->stranky = $stranky;
```

Obr. 4.4: Zend controller - zobrazení stránkovaných dat

Po vložení dat do tabulky jsou dvě možnosti, buď pomocí tlačítka *Hledat* dát vyhledat požadavek nebo tlačítkem *Smazat* smazat formulář. U každé buňky formuláře je ošetření pomocí validátorů a filtrů. Při zadání čísla do kolonky jména, dojde k jejímu odfiltrování. Po stisknutí tlačítka *Hledat* se zkontrolují vstupní data a zavolá se funkce *hledatClen* na obrázku 4.5 v modelu *Clen*. V této funkci se provede SELECT v tabulce *user_profiles*. Podmínky vyhledávání jsou zadány pomocí funkcí WHERE a LIKE, při porovnávání musí být za každou proměnnou procento, které nahradí libovolné znaky. Výsledky se zobrazí v původním okně pod formulářem. Každý uživatel má ve svém řádku odkaz na stránku, kde je možné upravit profil. Stisknutím tlačítka *Smazat* se před předáním do modelu data z formuláře vymažou a model vrátí všechny uživatele.

```
$db = Zend_Db_Table::getDefaultAdapter();
$select = $db->select()
                ->from('user_profiles')
                ->join('users', 'users.id = user_profiles.user_id')
                ->join('role', 'users.role = role.id_role')
                ->where('user_id LIKE ?',$user_id."%")
                ->where('jmeno LIKE ?', $jmeno."%")
                ->where('prijmeni LIKE ?', $prijmeni."%")
                ->where('prezdivka LIKE ?', $prezdivka."%")
                ->where('users.email LIKE ?', $email."%")
                ->where('mobil LIKE ?', $mobil."%")
                ->where('ulice LIKE ?', $ulice."%")
                ->where('cislop LIKE ?', $cislop."%")
                ->where('mesto LIKE ?', $mesto."%")
                ->where('psc LIKE ?', $psc."%")
                ->where('poznamka LIKE ?', $poznamka."%");
$data = $db->query($select)->fetchAll();
```

Obr. 4.5: Zend model - vyhledání uživatelů

4.3.5 Správa uživatelů

Správa uživatelů je část kódu, které se stará o to aby do stránek nevstoupil neoprávněně uživatel. Dále se stará o registraci, přihlašování a odhlašování uživatelů. Nejprve se musí uživatel zaregistrovat, po registraci se již může přihlásit. Zaregistrovaný uživatel získává v základu přidělená práva *guest*, který má stejná práva jako uživatel nepřihlášený. Při ukončení internetového prohlížeče se musí uživatel znovu přihlásit.

Registrace - Vytvoří se nový formulář, který se načte do okna registrace. Po přijetí validovaných a filtrovaných dat z registrace se odešlou do modelu *Clen* akce *registerClen* na obrázku 4.6, kde se data uloží do databáze. Proměnná *heslo1* se zašifruje pomocí funkce SHA1. Data v modelu jsou uložena do dvou databázových tabulek. Tabulka *users* slouží pro přihlašovací údaje a správu uživatele. Tabulka *user_profiles* slouží pro uložení osobních údajů uživatele.

```
public function registerClen($jmeno, $prijmeni, $prezdivka, $email, $mobil,
                  $ulice, $cislop, $mesto, $psc, $info, $poznamka, $heslo1)
{
  $data = array('password' => $heslo1,'email' => $email,
                'activated'=>1,'role' => '2','id_stav' => 201);
  $this->_db->insert('users',$data);
  $last=$this->_db->lastInsertId('users');
  $data = array('user_id'=> $last,'jmeno' => $jmeno,'prijmeni' => $prijmeni,
    'prezdivka' => $prezdivka,'mobil' => $mobil,'ulice' => $ulice,
    'cislop' => $cislop,'mesto' => $mesto,'psc' => $psc,
    'info' => $info,'poznamka' => $poznamka);
  $this->_db->insert('user_profiles',$data);
}
```

Obr. 4.6: Zend model - registrace uživatele

Login - Logování do stránek se provádí pomocí emailu a hesla. Pokud je již uživatel přihlášen, tak mu není umožněno se přihlásit znova, musí se nejprve odhlásit. Pokud jsou přijata data od uživatele, vytvoří se pomocí pomocné funkce *getAuthAdapter* objekt z knihovny pro autentizaci *Auth*, který se odkazuje na tabulku *users*, sloupce *email* a *password*, email musí být jedinečný proto se používá k určení uživatele a heslo k samotnému ověření uživatele. Pomocí metody *getInstance* se ověří zda zadané údaje odpovídají uživateli. Přístupová identita uživatele bez hesla se uloží do registru. Následuje přesměrování na úvodní stránku.

Logout - Odhlášení se provádí pomocí odkazu v pravém horním rohu. V controlleru *AuthController* akci *logout* na obrázku4.8. Odhlášení spočívá v tom, že se vymažou v registru údaje uživatele a přejde se na obrazovku s přihlášením.

```
function loginAction()
{if(Zend_Auth::getInstance()->hasIdentity())
 $this->_redirect('index/index');

 $request = $this->getRequest();
 $form = new Application_Form_Login();
 if($request->isPost()){
   if($form->isValid($this->_request->getPost())){//kontrola validace
     if($this->stav($form->getValue('email1'))==201){   //kontrola stavu
       $authAdapter = $this->getAuthAdapter();
       $email1 = $form->getValue('email1');
       $heslo1 = $form->getValue('heslolog');
       $heslo1=SHA1($heslo1);
       $authAdapter ->setIdentity($email1)
                    ->setCredential($heslo1);

       $auth = Zend_Auth::getInstance();
       $result = $auth->authenticate($authAdapter);
       if($result->isValid()){
          $identity = $authAdapter->getResultRowObject(null,'password');
          $authStorage = $auth->getStorage();
          $authStorage->write($identity); // uložení uživatele
          $this->_redirect('index/index');
       }
   }if($this->stav($form->getValue('email1'))==202){
       echo 'Účet je zablokován, napište správci!';}
    else{
       echo 'Heslo je špatně zadáno.'; }  ///vypis chyboveho hlaseni
    }
  }
   $this ->view->form = $form;
}
```

Obr. 4.7: Zend controller - přihlášení uživatele

```
function logoutAction()
    {
      Zend_Auth::getInstance()->clearIdentity();
      $this->_helper->redirector('login');
    }
```

Obr. 4.8: Zend controller - odhlášení uživatele

4.3.6 Přidělování rolí

Role určují kam má uživatel přístup povolen a co může dělat. Pokud uživatel není přihlášen má přidělenou roli *guest*, která může stránky pouze prohlížet. Po zaregistrování mu role *guest* zůstane, *admin* má právo roli členům měnit. K nastavení práv uživatelů jsem využil knihovnu *Acl*. K nastavení oprávnění jsem použil nastavení práv uživatelů skrz model, ukázka zadání práv viz. obrázek4.9. Touto metodou je nastavení rolí a jejich práv snadné a rychlé. Do modelu je potřeba vložit všechny role a zdroje. Při vytváření rolí jde přidat jako další parametr roli od které bude dědit

práva. Samotné přidělení práv se provádí pomocí *allow* a *deny*, které mohou roli povolit nebo zakázat přístup do kontroleru. Oprávnění lze rozšířit o povolení přístupu k samotným akcím. Nastavení práv tímto způsobem je vhodné pouze pro menší počet rolí a oprávnění. Nevýhodou je, že změna práv je možná pouze v souboru. Nelze dynamicky přidávat pomocí aplikace nové role a jejich oprávnění.

```
//////////////////////role/////////////////////////////////////
        $this->addRole(new Zend_Acl_Role('2'));  //guest
        $this->addRole(new Zend_Acl_Role('3'),'2'); //vedouci
        $this->addRole(new Zend_Acl_Role('4'),'3'); //hl.vedouci
        $this->addRole(new Zend_Acl_Role('1'));  //admin
//////////////////zdroje controllery//////////////////////////////
        $this->add(new Zend_Acl_Resource('auth'));
        $this->add(new Zend_Acl_Resource('index'));
        $this->add(new Zend_Acl_Resource('clen'));
        $this->add(new Zend_Acl_Resource('akce'));
        $this->add(new Zend_Acl_Resource('rubriky'));
        $this->add(new Zend_Acl_Resource('menu'));
//pravidla guest/////////////////////////////////////////////////
        $this->allow('2','auth', 'login');
        $this->allow('2','auth','register');
        $this->allow('2','auth','logout');
        $this->allow('2','menu','menuobsah');
        $this->allow('2','rubriky','rubrikyjedna');
        $this->allow('2','akce','akcejedna');
        $this->allow('2','index','jednastranka');
        $this->allow('2','index','index');
//vedouci////////////////////////////////////////////////////////
         $this->deny('3','auth', 'login');
         $this->deny('3','auth','register');
         $this->allow('3','auth','logout');
         $this->allow('3','index');
         $this->deny('3','index','smazat');
```

Obr. 4.9: Zend model - práva uživatelů

Do souboru *Bootstrap* jsem vložil funkci, která kontroluje přihlášeného uživatele a jeho roli. Funkce si načte pravidla přístupu s modelu *Acl*, a pokud je uživatel přihlášen zjistí jeho roli. V závěru načte plugin *AccessCheck*, tento plugin proběhne před každým voláním kontroleru a v případě neoprávněného vstupu, přesměruje na funkci přihlášení. Ošetření zobrazení odkazů přístupu Zend funkce *isAllowed*, kde se zadá požadavek, jaká role a co za funkci chce použít, zadané údaje se porovnají podle pravidel. Ukázka ošetření view na obr. 4.10 je ze souboru *strankyseznam*, do kterého mají přístup *vedoucí*, *Hl.vedouci* a *admin*.

```
<?php $acl = new Application_Model_ACL();
if($acl->isAllowed(Zend_registry::get('role'),'index','pridat')){ ?>

<p><a href="<?php echo $this->url(array('controller'=>'index',
'action'=>'pridat'));?>">přidat stránku</a></p>

<?php } ?>
```

Obr. 4.10: Zend view - povolení zobrazení

4.3.7 Odesílání Emailů

Ukázka odesílání emailů je použita při vkládání nových stránek. Všichni uživatelé, kteří zaškrtnou *Zasílat aktuality*, budou při vložení nové stránky dostávat obsah této stránky na email. Pomocí funkce Mercury jsem si v programu XAMPP vytvořil dva emaily na localhostu, které používám ke zkoušení odesílání emailů. V controlleru *Index*, který slouží pro práci se stránkami, jsem do akce *pridat*, jak je vidět na obrázku4.11, přidal kód pro odesílání emailů. Nejdříve zavolám model *Mail*, který vrátí emailové adresy všech uživatelů požadujících zasílat aktuality. Vytvořím nový email kódovaný ve formátu UTF-8, pro správné zobrazování háčků a čárek. Do obsahu emailu vložím text a do předmětu nadpis stránky. Odesílání provádím v cyklu, kdy do odesílatele přidám vždy jednoho příjemce, před odesláním dalšího emailu vždy příjemce vymažu. Je nežádoucí aby příjemci viděli adresy jiných příjemců.

```
$config = array( // nastavení přihlášení k mailu
   'username' => 'test1@localhost',
   'password' => 'test1',);
 $tr = new Zend_Mail_Transport_Smtp('localhost',$config);
 $mail = new Application_Model_Mail(); //vytvoření objektu
 $email=$mail->sendMail(); //získání emailů k odeslání
 $mail = new Zend_Mail('UTF-8');
 $mail->setBodyText($text);   //nastavení obsahu
 $mail->setSubject($nazev);
 $mail->setFrom('test1@localhost','Testovací email ZEND');
 foreach($email as $send){
 $mail->clearRecipients(); //smazání předchozího příjemce
 $mail->addTo($send['email']); // vložení příjemce
 $mail->send($tr); //odeslání emailu
 }
```

Obr. 4.11: Zend controller - odesílání emailu

4.4 CodeIgniter

4.4.1 Konfigurace

Konfigurace u CodeIgniter se provádí ve složce config jak je zobrazeno v adresářové struktuře viz. obrázek3.3. Konfigurace je rozdělena do více souborů. Nastavil v souboru autoload.php načítání používaných knihoven a helperů. V souboru config.php URL adresu, vstupní obrazovku, která se má otevřít při prvním spuštění a kódování stránek. V souboru email.php základní nastavení pro odesílání emailů. V souboru database.php nastavení přístupu do databáze a kódování databáze, nastavení viz. obrázek4.12.

```
$active_group = 'default';
$active_record = TRUE;

$db['default']['hostname'] = 'localhost';
$db['default']['username'] = 'root';
$db['default']['password'] = '';
$db['default']['database'] = 'skautcz';
$db['default']['dbdriver'] = 'mysql';
$db['default']['dbprefix'] = '';
$db['default']['pconnect'] = TRUE;
$db['default']['db_debug'] = TRUE;
$db['default']['cache_on'] = FALSE;
$db['default']['cachedir'] = '';
$db['default']['char_set'] = 'utf8';
$db['default']['dbcollat'] = 'utf8';
$db['default']['autoinit'] = TRUE;
$db['default']['stricton'] = FALSE;
```

Obr. 4.12: CodeIgniter konfigurace databáze

4.4.2 Filtrování a ošetření vstupních dat

Tyto funkce jsou důležité při vkládání vstupních dat od uživatelů. Využívají se hlavně k zamezení vložení špatných dat. K zamezení poškození celého systému programu. Kontrolují zda uživatel vložil data správně. Do pravidel validace se zadává jméno proměnné, název který se bude vypisovat při špatném zadání v chybových hlášení. Všechny pravidla zapsány za sebou odděleny |. Na obrázku4.13 je ukázána vstupní validace při registraci uživatele. Validace se nastavuje a provádí v controlleru. V základu neumí validace pracovat s háčky a čárkami. Musel jsem si vytvořit vlastní validátor. Tento validátor se nachází
v application\libraries\MY_Form_validation.

- required - Vyžaduje naplnění proměnné daty.

- is_unique [*users.email*] - Zkontroluje zda parametr je v tabulce jediný. Povinné parametry tabulka a sloupec.
- min_length[4] a max_length[12] - Udávají počet vstupních znaků minimální a maximální počet, tyto validátory lze používat samostatně.
- is_natural - Data musí být pouze přirozená čísla.
- valid_emails - Vložená data musí být emailová adresa.
- trim - Provede ořezání, které odfiltruje bílé znaky na konci a na začátku.
- xss_clean - Odstraní škodlivé údaje. Tento filtr lze nastavit přímo v konfiguraci, všechny data předávaný pomocí GET, POST nebo COOKIES budou kontrolovány a filtrovány.

```
$this->form_validation->set_rules('jmeno','Jméno', 'trim|alpha|required|xss_clean|max_length[30]');
$this->form_validation->set_rules('prijmeni','Příjmení', 'trim|alpha|required|xss_clean|max_length[30]');
$this->form_validation->set_rules('email', 'Email', 'trim|required|xss_clean|
valid_email|is_unique[users.email]');
$this->form_validation->set_rules('password', 'Heslo', 'trim|required|xss_clean|
min_length['.$this->config->item('password_min_length', 'tank_auth').']|
max_length['.$this->config->item('password_max_length', 'tank_auth').']|alpha_dash');
$this->form_validation->set_rules('prezdivka', 'Přezdívka','trim|alpha|xss_clean|max_length[30]');
$this->form_validation->set_rules('mobil', 'Mobilní číslo','trim|xss_clean|is_natural|min_length[9]');
$this->form_validation->set_rules('ulice', 'Ulice','trim|xss_clean');
$this->form_validation->set_rules('cislop', 'Číslo popisné','trim|xss_clean|is_natural');
$this->form_validation->set_rules('mesto', 'Město','trim|xss_clean|alpha|max_length[50]');
$this->form_validation->set_rules('psc', 'PSČ','trim|xss_clean|is_natural');
$this->form_validation->set_rules('info', 'info','trim|xss_clean');
$this->form_validation->set_rules('poznamka', 'Poznámka','trim|xss_clean');
```

Obr. 4.13: CodeIgniter validace user

4.4.3 Stránkování

Stránkování využívám při zobrazování stránek. CodeIgniter nabízí knihovnu *Pagination*,pro usnadnění práce při tvorbě stránkování. Při nastavení stránkování v controlleru je nejdříve nutné načíst používanou knihovnu, následně nastavit stránkování. Je potřeba nastavit adresu na tu, ze které se stránkování bude provádět, protože na konci adresy si CodeIgniter ukládá ukazatelna první prvek na stránce. Další, co je potřeba nastavit, je celkový počet prvků, počet zobrazených prvků na jedné stránce, počet odkazů před a po aktuální stránce a povolit nebo zakázat zobrazení odkazů na první a poslední stránce. Z controlleru volám model, do kterého předávám počet prvku na stránce a počáteční prvek z aktuální obrazovky($offset$). V modelu načtu všechny stránky postupně podle id a vrátím pouze potřebný počet posunutý o offset. Data vrácená z modelu pošlu do view, kde pod vložená data přidám odkazy sloužící pro listování mezi stránkami. Na obrázku4.14 je zobrazeno nastavení stránkování v controlleru *stranky* v akci *strankyseznam*.

```
$this->load->library('pagination');
$config = array(
        'base_url' => base_url().'index.php/stranky/strankyseznam/',
        'total_rows' => $this->db->count_all_results('stranky'),
        'per_page' => 10,      // pocet radku
        'first_link' => FALSE,    //'prvni',
        'last_link' => FALSE, //'posledni',
        'num_links' => 10,    //pocet cisel na kazdu stranu
        'display_pages' => TRUE, //false= pouze sipky bez cisel
);
$this->pagination->initialize($config);
```

Obr. 4.14: CodeIgniter controller stránkování

4.4.4 Vyhledávání

Vyhledávání je využito u vyhledávání uživatelů. Do vyhledávání uživatelů se dá dostat pomocí odkazu v horní liště na stránce se seznamem stránek. Na táto stránce jsou spojeny dva bloky, a to formulář pro zadávání požadavku hledání a pod ním následuje tabulka s výsledky. Všichni uživatelé se zobrazí při prvotním zobrazení, nebo pokud není žádná omezující podmínka. Na buňky formuláře je nastavena validace, k vypsání chyby dojde například při vložení čísla do jména, zároveň nedojde k vyhledání a ve výsledku jsou všichni uživatelé. Samotného vyhledávání, po zadání dat do formuláře a odeslání formuláře pomocí tlačítka *Hledat*, se provede validace a data se odešlou do modelu *user_model*, který je zobrazen na obrázku4.15. V modelu pomocí funkce SELECT vyhledám všechna potřebná data z tabulky *user_profiles*, odeberu data, která nejsou stejné s formulářem pomocí WHERE a data neobsahující data z formuláře pomocí funkce LIKE. Všechny vyhledané uživatele seřadím podle *id*. Z modelu vracím pole obsahující výsledek a počet výsledků. Výsledky se zobrazí v původním okně, za každým uživatelem je tlačítko pro úpravu profilu. Stisknutím tlačítka *Vymazat* se předávaná data smažou a vrátí se prázdný formulář.

4.4.5 Správa uživatelů

Správa uživatelů funguje stejně jako u Zend, do souborů CodeIgniter jsem nahrál již vytvořenou knihovnu [19]. Tato knihovna je vybavena vlastním šifrováním hesel, které jsem nepoužil z důvodu shody s hesly Zend. U knihovny jsem vypnul aktivaci pomocí emailu. Dále jsem vypnul zadávání captcha při přihlašování, kterou tato knihovna nabízí. Knihovna nabízí automatické přihlašování, doba přihlášení je nastavena v konfiguračním souboru *tank_auth*. V knihovně jsou předpřipraveny obrazovky pro přihlášení, registraci a změnu údajů. Stávající databázi bylo nutné rozšířit o další tabulky, které knihovna využívá *users*, *ci_sessions*, *login_attempts* a *user_autologin*. Tabulku *users* využívají oba frameworky pro správu uživatelů.

```
  $query = $this->db->select('*');
  $query = $this->db->from('user_profiles');
  $query = $this->db->join('users','users.id = user_profiles.user_id');
  if($profile_data['user_id'])
  $query = $this->db->where('user_id',$profile_data['user_id']);
  $query = $this->db->like('jmeno',$profile_data['jmeno']);
  $query = $this->db->like('prijmeni',$profile_data['prijmeni']);
  $query = $this->db->like('prezdivka',$profile_data['prezdivka']);
  $query = $this->db->like('mobil',$profile_data['mobil']);
  $query = $this->db->like('ulice',$profile_data['ulice']);
  $query = $this->db->like('cislop',$profile_data['cislop']);
  $query = $this->db->like('mesto',$profile_data['mesto']);
  $query = $this->db->like('psc',$profile_data['psc']);
  $query = $this->db->like('poznamka',$profile_data['poznamka']);
  $query = $this->db->like('users.email',$profile_data['email']);
  $query = $this->db->order_by('users.id');
  $query = $this->db->get('');

  $user['user']= $query->result();
  $user['rows']= $query->num_rows();
return $user;
```

Obr. 4.15: CodeIgniter model - vyhledání uživatelů

4.4.6 Přidělování rolí

Pokud není uživatel přihlášen, má přidělenou roli *guest*. Při registraci je každému uživateli ponechána role *guest*. Jinou roli přidělí *admin*.

Práva rolím jsou přidělena v databázi, v tabulce *zdroje* jsou všechny přístupy a v tabulce *prava* jsou pravidla. Pomocí helperu *auth_helper* funkce *prava*, která je na obrázku4.17, se určuje, zda má uživatel povolený přístup. Pomocí selectu zjistím, zda-li je pro uživatele povolen přistup. Pokud má uživatel přístup, zobrazí se mu tlačítko ke stisknutí. Další funkcí je *opravneni*, tato funkce volá *prava*, pokud je přístup zamítnut, funkce přesměruje požadavek na úvodní stránku, tímto je ošetřeno zavolání přes adresu controlleru.

```
if(prava('clen hledat')){
  echo anchor('clen/index','Uživatelé');
  echo '  ';}
if(prava('akce index')){
  echo anchor('akce/index','Akce');
  echo '  ';}
if(prava('stranky seznam')){
  echo anchor('stranky/strankyseznam','Stránky');
  echo '  ';}
```

Obr. 4.16: CodeIgniter view - povolení zobrazení odkazu

```php
function prava($prava= null){
  $uid=uid(); //zjisti zda má uživatel oprávnění
  $ci =& get_instance();
  $query = $ci->db->select('zdroj.kontrolerakce');
  $query = $ci->db->from('prava');
  $query = $ci->db->join('zdroj','zdroj.id_zdroj = prava.zdroj');
  if($uid){
  $query = $ci->db->join('users','users.role = prava.role');
  $query = $ci->db->where('users.id',$uid);
  }else
  $query = $ci->db->where('prava.role','2');
  $query = $ci->db->where('zdroj.kontrolerakce',$prava);
  $query = $ci->db->get('');
  return $query->num_rows()?TRUE:FALSE;
}
function opravneni($prava= null){
     $vysledek=prava($prava);
     if(!$vysledek)
     redirect ('');
}
```

Obr. 4.17: CodeIgniter helper - práva a oprávnění

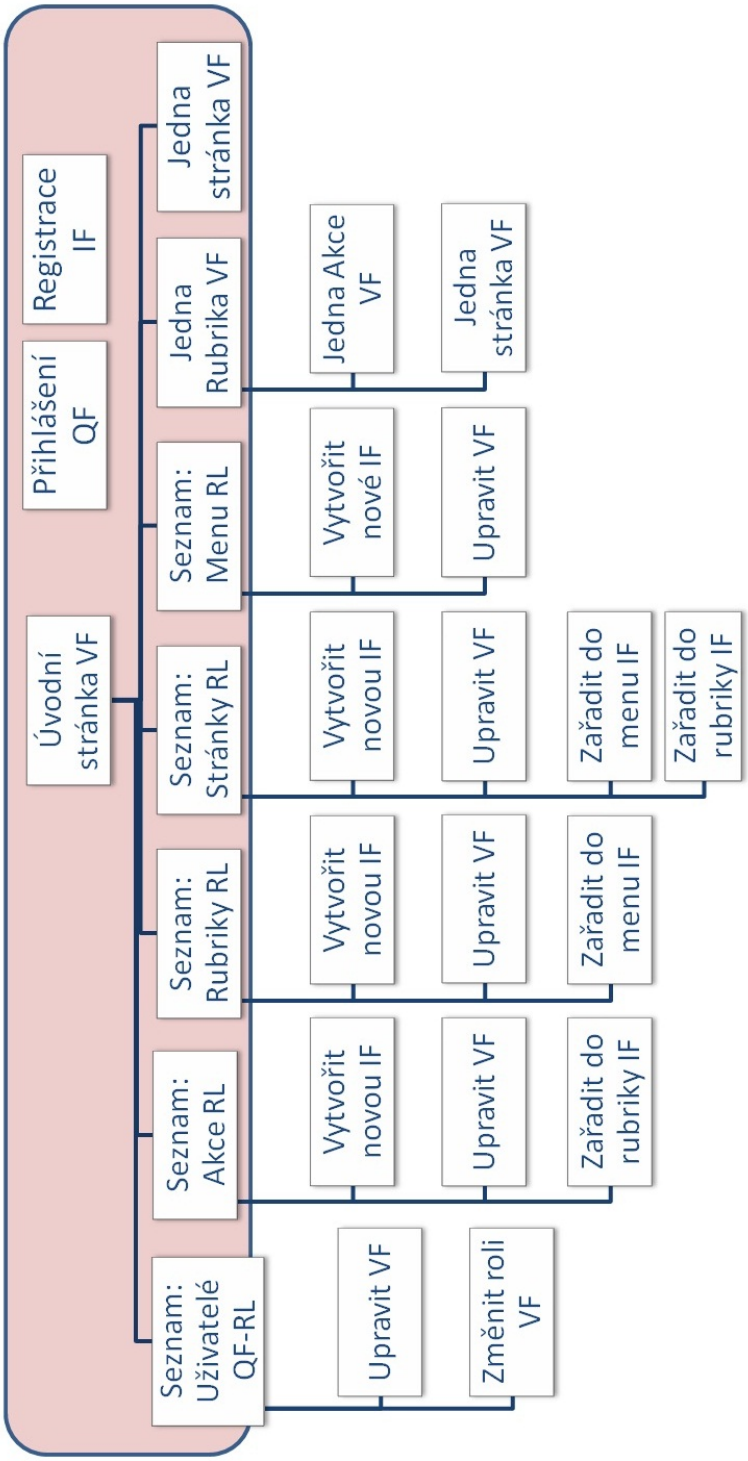
4.4.7 Odesílání Emailů

Odesílání emailů funguje stejně jako u stránek v Zend. V controlleru *Stranky* jsem do funkce *add* vložil volání na funkci *infomail*, která je vidět na obrázku4.18. Nejdříve z modelu *infomail* načtu všechny uživatele, kterým má být email odeslán. Nastavím jméno a adresu odesílatele. Cyklicky vkládám jména uživatelů a jejich emailové adresy a rozesílám aktualitu. Adresa a jméno odesílatele lze nastavit v konfiguraci.

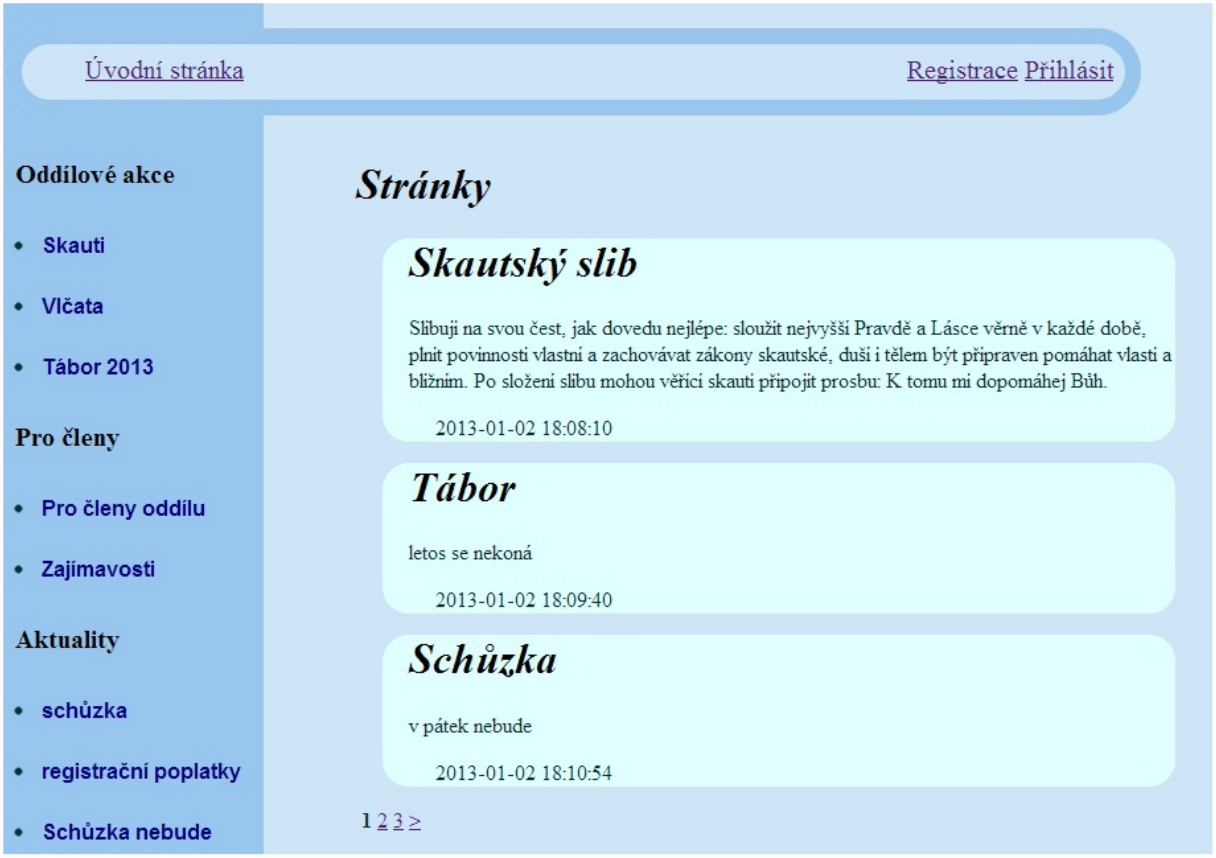
```php
function infomail($message)
      {opravneni('stranky infomail');
      $mail1 = $this->infomail_model->sendmail();
      $this->load->library('email');
      $this->email->from('test2@localhost', 'Testovaci email CodeIgniter');
      foreach($mail1 as $mail) :
      $this->email->to($mail->email, $mail->jmeno." ".$mail->prijmeni);
      $this->email->subject($message['nazev']);
      $this->email->message($message['text']);
      $this->email->send();
      endforeach;
      }
```

Obr. 4.18: CodeIgniter funkce infomail

5 USPOŘÁDÁNÍ VZOROVÝCH APLIKACÍ



Obr. 5.1: Navigace mezi obrazovkami



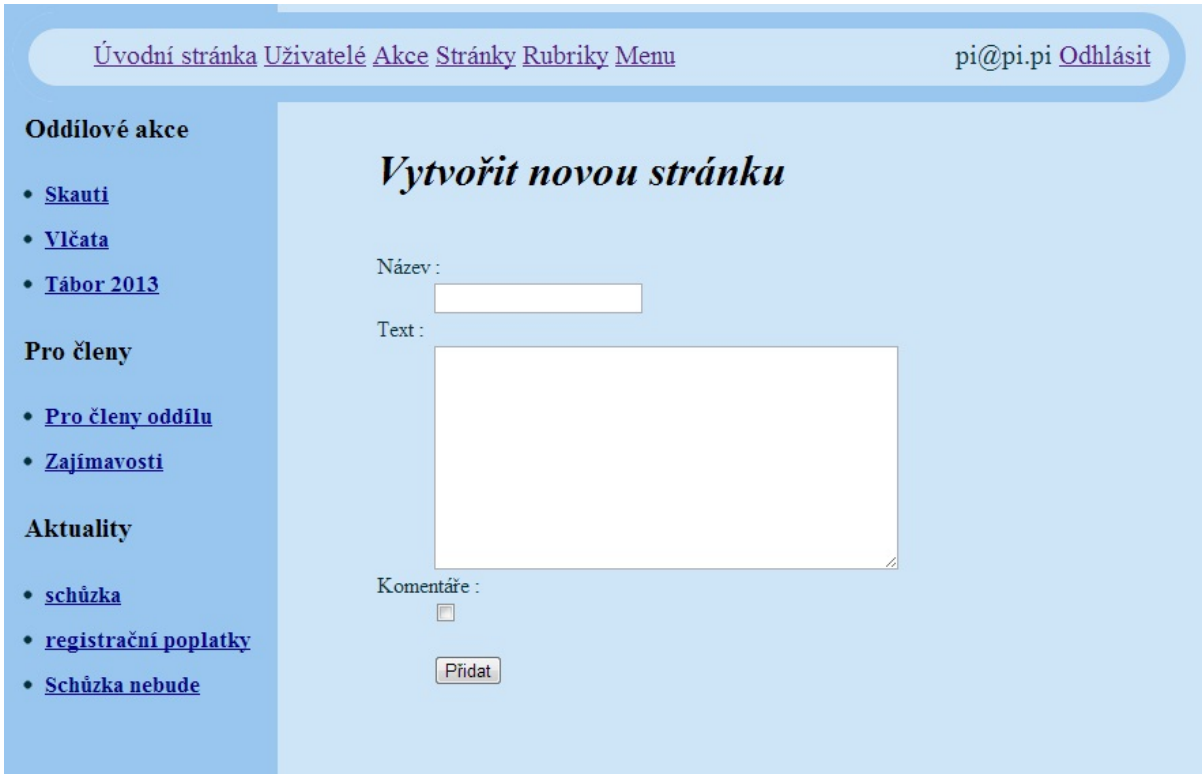
Obr. 5.2: Úvodní stránka v Zend(horní) a CodeIgniter(dolní)

Navigace mezi obrazovkami

Samotná navigace je zobrazena na obrázku5.1. Každá obrazovka je popsána názvem a typem obrazovky. Obrazovky, které jsou podbarvené, jsou dostupné z horního nebo z levého menu. V navigaci jsou zobrazeny všechny obrazovky. Do všech obrazovek má přístup pouze administrátor.

Grafický vzhled

Na obrázku5.2 je zobrazena úvodní obrazovka Zend a CodeIgniter. Formulář k vytvoření nové stránky je na obrázku5.3. Další obrázky vzorové aplikace jsou v příloze.

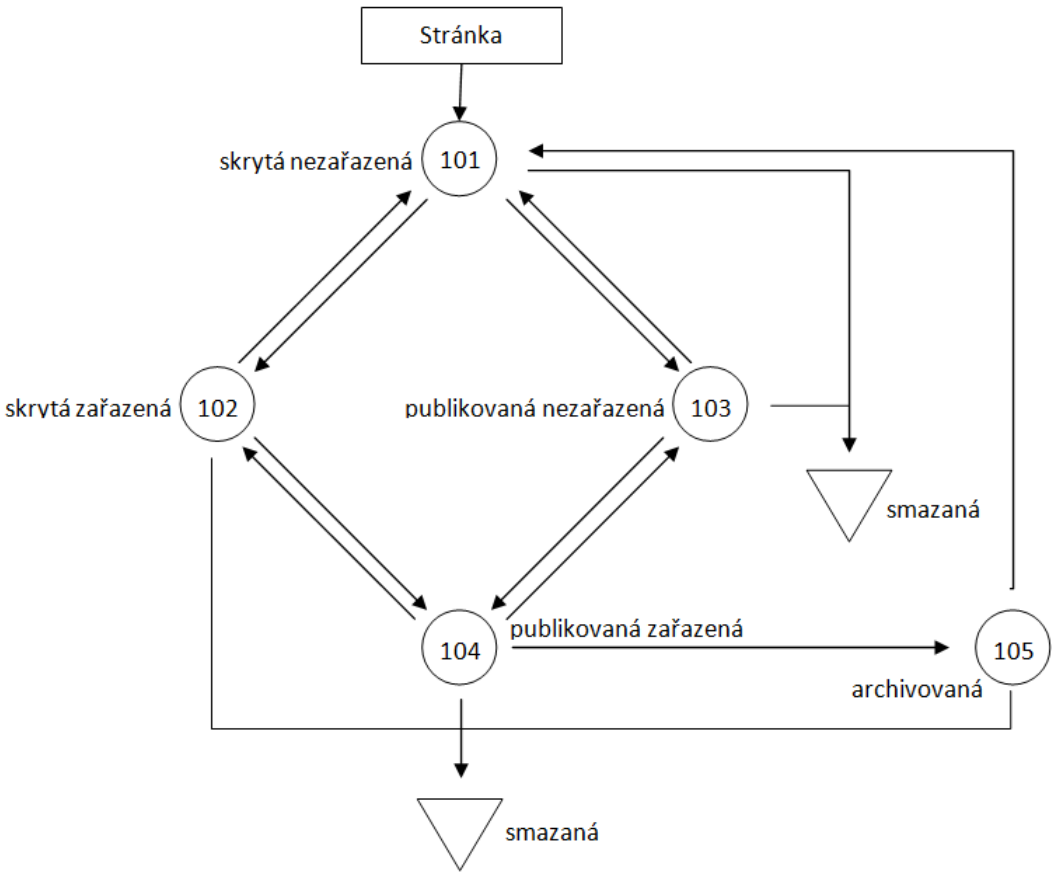


Obr. 5.3: Formulář k vytvoření nové stránky

6 ŽIVOTNÍ CYKLY

Každá entita, která má využívat životní cykly, má od samého vytvoření přidělen určitý stav.

Ke každému stavu lze přidělit vlastnosti týkající se stavů následujících, oprávněné role mají v daném stavu prvek vidět a moci s ním pracovat. Dle zvolené role může uživatel měnit stav prvků. Životní cykly se starají o entity Akce, Stránka, Rubrika a User. Entita User má jednoduchý životní cyklus, který má pouze dva stavy.



Obr. 6.1: Životní cyklus stránky

6.1 Popis životního cyklu

stav 101 - Stránka skrytá nezařazená
Při vytvoření nové stránky, je stránce přiřazen tento stav, odpovídá nezařazené a nepublikované stránce. Stránka je vidět pouze v záložce „Stránky“, do který má přístup „Vedouci“, „Hl.vedouci“ a „admin“. Zde je stránka možno měnit. Ze stavu 101 jsou dvě možnosti, co se stránkou udělat, všechny tři role můžou stránku zařadit do menu nebo do rubriky, což je stav 102. Další možnost mají „Hl.vedouci“ a „admin“, kteří mohou dát stránku publikovat, stránka by byla publikována, ale nezařazena.
stav 102 - Stránka skrytá zařazená
Stránka je zařazena do rubriky nebo do menu, je ji možno zpětně vyřadit a dostat se opět do stavu 101. „Hl.vedouci“ a „admin“ mohou z tohoto stavu stránku publikovat, čímž se dostane do stavu 104.
stav 103 - Stránka publikovaná nezařazená
Stránka je publikována, není však ještě zařazena do rubriky nebo menu. Z toho stavu mohou všechny předem zmiňované role stránku skrýt, nebo přiřadit do rubriky či menu.
stav 104 - Stránka publikovaná zařazená
Stránka je publikována a zařazena do rubriky nebo menu. V tomto stavu je stránka přístupná pro nepřihlášené uživatele, mohou se k ní dostat skrze postranní menu a příslušné rubriky. Oprávněné role mohou stránku skrýt, vyřadit z menu nebo z rubriky a archivovat. Archivováním se stránka odebere z rubriky nebo z menu a skryje se.
stav 105 - Stránka archivovaná
Archivované stránky je možno znovu obnovit, čímž se nastaví do počátečního stavu, do stavu 101.

Ze všech zvolených stavů lze u stránky upravovat obsah. Role „Hl.vedouci“ a „admin“ mohou stránky vymazat nezávisle na tom, v jakém stavu se nachází.

6.2 Tabulka stavů a přechodů Stránky

Tab. 6.1: Tabulka stavů Stránky

Id_stav	**Název**	**Zobrazená**
101	Stránka skrytá nezařazená	NE
102	Stránka skrytá zařazená	NE
103	Stránka publikovaná nezařazená	NE
104	Stránka publikovaná zařazená	ANO
105	Stránka archivovaná	NE

„Zobrazená“ - Ano znamená, zda je stránka přístupná v levém menu, jenž vidí všichni uživatelé bez ohledu na přihlášení.

Tab. 6.2: Tabulka přechodů Stránky

ID_stav		**Popis přechodu**	**Role**		
Z	**DO**		**Vedouci**	**Hl.vedouci**	**Admin**
	101	Vytvoření	ANO	ANO	ANO
101	102	Zařazení do rubriky/menu	ANO	ANO	ANO
101	103	Publikování	NE	ANO	ANO
102	101	Vyřazení z rubriky/menu	ANO	ANO	ANO
102	103	Publikování	NE	ANO	ANO
103	104	Zařazení do rubriky/menu	ANO	ANO	ANO
103	101	Skrytí	ANO	ANO	ANO
104	102	Skrytí	ANO	ANO	ANO
104	103	Vyřazení z rubriky/menu	ANO	ANO	ANO
104	105	Archivování	ANO	ANO	ANO
105	101	Obnovit	ANO	ANO	ANO
101		Smazat	NE	ANO	ANO
102		Smazat	NE	ANO	ANO
103		Smazat	NE	ANO	ANO
104		Smazat	NE	ANO	ANO
105		Smazat	NE	ANO	ANO

7 POROVNÁNÍ VZOROVÝCH APLIKACÍ

K vytvoření srovnávacích aplikací byly využity tyto dva frameworky, a to Zend framework 1.12.3 a CodeIgniter 2.1.3. Pro porovnání aplikací využiji stejné kategorie jako při popisu částí aplikací v kapitole 4.3.

7.1 Filtrování a ošetření vstupních dat

Filtrování a validace vstupních dat je důležitým prvkem stránek a využívá se ve všech stránkách, kde má uživatel umožněno vkládat data. Oba zvolené frameworky validaci a filtraci mají a disponují širokou škálu validátorů a filtrů. Zatím co CodeIgniter nabízí pouze základní validátory, Zend nabídl více validátorů, které nejsou často používané. U vzorové aplikace jsem si vystačil s validátory od obou frameworků, nemusel jsem vytvářet vlastní validátory.

U **Zend** se využívá formulářů, ve kterých se nastaví pro jednotlivé elementy vlastnosti zvlášť. Výsledný formulář se předává do view, kde se vytiskne jedním příkazem. Všechny zobrazení formulářů mají skoro stejné view. V kontroleru se provede kontrola všech validátorů a filtrů. Validátory fungovali i s háčky a čárkami. Chybová hlášení si Zend volá přímo z knihoven. Přepis do českého jazyka byl zásahem do knihovny. Každý validátor měl chybovou hlášku ve svém souboru. Pro zobrazení o jaký prvek se jedná, jsem si do validátorů přidal vlastní texty. Data jsem předával jednotlivě modelu, čímž jsem si zkomplikoval práci, musel jsem pro rozdílná vstupní data tvořit vždy novou funkci v modelu.

U **CodeIgniter** se žádné formuláře nevyužívají, zobrazení samotného formuláře se vytváří přímo ve view, tam jsem nastavil parametry vstupních elementů. Validátory se nastavují přímo v kontroleru a narozdíl od Zend mi přišlo zadávání jednoduší a přehlednější. Validátory se zadávaly do jednoho řádku a stačilo zadat pouze název validátoru a dodatečný parametr, pokud byl nějaký potřeba. U validátorů jsem přidával název proměnné, který byl součástí chybového hlášení. Chybové hlášení se stala přehledná a na první pohled bylo vidět, kde chyba nastala. Zaujala mě validace *is_unique*, která zadáním názvu tabulky a sloupce ošetřuje unikátnost dat. Texty chybových hlášení jsou umístěny v jednom souboru, bylo nutné je přepsat do českého jazyka. Při validaci, pokud se jednalo pouze o text, jsem narazil na problém, CodeIgniter nepodporoval háčky a čárky. Musel jsem si rozšířit původní validátor o zvolená písmena. Data jsem do modelu předával jako pole, čímž jsem si ušetřil psaní samostatných funkcí v modelu.

Pro vytvoření jednoho elementu s validátory jsem u Zend potřeboval napsat 9 řádků kódu a CodeIgniter 11 řádků, formulář u CodeIgniter byl již uspořádán podle mých

představ, u Zend byly elementy pod sebou, uspořádal jsem je pomocí CSS stylu.

7.2 Stránkování

S touto funkcionalitou nabízí pomoc oba nástroje.
U **Zend** se v kontroleru vytvoří objekt stránkování a nastaví se dílčí parametry stránkování. Objekt stránkování s daty se předají do view. Přijatá data se zobrazí, v mém případě volám další view, ve kterém mám vykreslení navigace stránkování. Zde je možno nastavit vlastní formát prvků navigace.
U **CodeIgniter** se konfigurace stránkování provádí podobně jako u Zend. Parametry stránkování se nastaví v kontroleru a s daty se předají do view. Ve view se zobrazí data a k vytvoření poslouží jeden příkaz, který navigaci mezi stránkami celý vytvoří. Stránkování je jednoduchým prvkem u Zend jsem stránkování vytvořil pomocí 25 řádků k vytvoření stránkování u CodeIgniter jsem si vystačil s 11 řádky kódu. Zend nabídl vlastní formátování navigace stránkování.

7.3 Správa uživatelů

O správu uživatelů se v Zend starají autentizační knihovna *auth* a knihovna *acl*, ta se stará o práva uživatelů. CodeIgniter v základu nenabízí žádnou knihovnu, co by se o správu uživatelů starala. CodeIgniter jsem si rozšířil o volně dostupnou knihovnu *Tank_auth*, jejíž stručný popis je v kapitole2.2.1.
U **Zend** při práci s uživateli využívám Bootstrap.php soubor, který proběhne před spuštěním každé akce. Z něho volám plugin AccesCheck, který kontroluje, zda ke zvolené akci má uživatel přístup. Model Acl slouží k nastavení uživatelských oprávnění, všechna práva jsou nastavena v tomto souboru, tato metoda nastavení práv je jednoduchá, avšak nevhodná pro velké a složité aplikace. Pro přihlášení a komunikaci s uživatelem používám kontroler AuthControler. Po přihlášení jsou data uživatele uložena v registru.
U **CodeIgniter** jsem si nahrál knihovnu *Tank_auth* do své aplikace a upravil pro své potřeby. Auth knihovna se stará o přihlašování, registraci a odhlašování. Knihovna využívá tabulky, které jsem přidal do své databáze. Informaci o přihlášeném uživateli si ukládá do session. K práci s rolemi jsem vytvořil auth_helper, který se stará o oprávnění přístupu uživatele. V každá akci je vloženo volání funkce, která zkontroluje oprávnění přístupu. Ve view je u každého odkazu podmínka podmíněna právy přístupu.
Na rozdíl od CodeIgniter mi Zend nabídl knihovny pro správu uživatelů, v tomto porovnání dle mého názoru je Zend jasně lepší. Budu-li srovnávat knihovny Zend

a vloženou knihovnu, která měla již kontrolery a view vytvořeny a se svými vlastnostmi již hotového prostředí určitě vyhrává knihovna pro CodeIgniter. Velkým plusem je u Zend kontrola přístupu rolí, akce v kontroleru jsou automaticky ošetřeny proti neoprávněnému použití, aniž by byl přidán kód k zamezení přístupu.

7.4 Odesílání Emailů

Odesílání hromadných emailů a hlavně automatiky, je velkým ulehčením práce. Odesílání emailu je ukázáno při vkládání stránky.
U **Zend** je nastavení jednoduché a rychlé. Nastaví se údaje od koho email je, komu má být doručen, předmětu a samotný obsah.
U **CodeIgniter** bylo potřeba nastavit několik parametrů v config. Odesílání mailu je stejně složité jako u Zend.
U Zend bylo k odeslání mailu zapotřebí napsat 14 řádků kódu u CodeIgniter 9 řádků kódu, odeslání mailu je v obou prostředích stejně jednoduché.

7.5 Rychlost

Rychlost odezvy stránek je velmi důležitá stejně jako vzhled stránek, který jsem u vzorových aplikací neřešil. Rychlost odezvy stránek jsem měřil pomocí aplikace Developer Tools, která je součásti prohlížeč Google Chrome. Vzorové aplikace jsem testoval na osobním počítači s parametry:

- procesor: Intel Core2 Duo T7100 1,80 GHz
- paměť: 2 GB
- HDD: 160 GB 5400 rpm

Protředí:

- Apache 2.4.3
- MySQL 5.5.27
- PHP 5.4.7

Měření rychlosti bylo pro každé měření provedeno desetkrát vždy pro jednu aplikaci, následně byl prohlížeč vypnut a změřil jsem odezvu druhé aplikace. Pro porovnání odezvy jsem si zvolil měření doby odezvy při načtení úvodní stránky, kdy nebyl žádný uživatel přihlášen. Další měření bylo pro přihlášení uživatele, vyhledání uživatele, publikování stránky a otevření rubriky v postranním menu.

Tab. 7.1: Porovnání rychlosti - úvodní stránka

	vzorků	**průměr** [s]	**MIN** [s]	**MAX** [s]
Zend	10	1,415	1,320	1,520
CodeIgniter	10	0,182	0,167	0,198
Procentuální rozdíl	10	**12,86 %**	**12,65 %**	**13,03 %**

Tab. 7.2: Porovnání rychlosti - přihlášení

	vzorků	**průměr** [s]	**MIN** [s]	**MAX** [s]
Zend	10	1,415	1,320	1,520
CodeIgniter	10	0,141	0,116	0,169
Procentuální rozdíl	10	**10,66 %**	**8,92 %**	**12,52 %**

Tab. 7.3: Porovnání rychlosti - vyhledání uživatele

	vzorků	**průměr** [s]	**MIN** [s]	**MAX** [s]
Zend	10	1,447	1,420	1,480
CodeIgniter	10	0,169	0,152	0,182
Procentuální rozdíl	10	**11,66 %**	**10,70 %**	**12,30 %**

Tab. 7.4: Porovnání rychlosti - publikování stránky

	vzorků	**průměr** [s]	**MIN** [s]	**MAX** [s]
Zend	10	1,296	1,250	1,350
CodeIgniter	10	0,121	0,112	0,136
Procentuální rozdíl	10	**9,35 %**	**8,96 %**	**10,07 %**

Tab. 7.5: Porovnání rychlosti - otevření rubriky

	vzorků	**průměr** [s]	**MIN** [s]	**MAX** [s]
Zend	10	1,330	1,290	1,360
CodeIgniter	10	0,161	0,144	0,174
Procentuální rozdíl	10	**12,11 %**	**11,16 %**	**12,79 %**

7.6 Velikost

Pro porovnání jsem vzal i velikost samotné aplikace. Počet řádků kódu, který byl potřeba napsat.

Tab. 7.6: Velikost aplikací

framework	**velikost [kB]**	**počet řádků kódu[-]**
Zend	126	3000
CodeIgniter	293	2600

U CodeIgniter není započten kód, který náleží knihovně Takn_auth(přibližně 1000 řádků). Rozsah napsaného kódu se znatelně neliší.

7.7 Zhodnocení

Vytváření validace vstupních dat u CodeIgniter byla jednodušší a přehlednější. Ke zvolenému elementu se přidávají validátory jako parametr. Pozitivní věcí u Zend je validace pracující s háčky a čárkami. U překladu chybových hlášení je vhodné umístění všech hlášení do jednoho souboru, jak to měl nastaveno CodeIgniter. Součástí validátoru u CodeIgniter byl název proměnný,který se vypisoval s chybou.
Nastavení odesílání emailů bylo u obou nástrojů stejně složité. U CodeIgniter mi schází možnost přijímání emailů.
K vytvoření stránkování u CodeIgniter potřebuje vývojář napsat polovinu kódu oproti Zend. Kód, o který je stránkování v Zend větší, slouží k vytvoření navigace mezi stránkami. CodeIgniter navigaci vygeneruje.
Pomoc při práci s uživateli poskytuje pouze Zend. Knihovny Zend jsou provázané s celým programem, nebylo nutně nastavovat kontrolu, zda-li má uživatel právo vykonat akci. Knihovny u CodeIgniter bylo nutné rozšířit o přídavnou knihovnu, která se starala pouze o autentizaci. Správu přístupu rolí podle práv přídavná knihovna nenabídla.
Pro porovnání jsem porovnal počet napsaných řádků kódu u obou aplikací, hodnoty jsou zobrazeny v tabulce7.6. Vývojová prostředí mi byli nápomocná stejně. Rychlost odezvy aplikací je zobrazena v tabulkách 7.1 až 7.5. Všechny změřené hodnoty jsou v přílozeE.1. Z naměřených hodnot je vidět, že odezva aplikace napsané v programu CodeIgniter se vykoná desetkrát rychleji než aplikace v programu Zend. Při běžném používání Zend se pro zrychlení používá ukládání do dat do mezipaměti což aplikaci urychlí.

Zend oproti CodeIgniter nabízí souborový typ form, které slouží k vytvoření celého těla formuláře, tento formulář se vkládá do view, usnadní práci pokud využíváme stejný formulář ve vice odlišných zobrazeních.
Pro porovnání jsem vlastnostem aplikace přidělil bodové ohodnocení od 0 bodů do 5 bodů. K de hodnota 5 bodů je nejlepší a hodnota 0 je nehorší.

Tab. 7.7: Vyhodnocení frameworků

Vlastnosti aplikací	**Zend**	**CodeIgniter**
Validace	3	4
Email	4	2
Správa uživatelů	5	0
Stránkování	3	3
Struktura aplikace	5	4
Rychlost odezvy	0	5
Součet	**20**	**18**

Z porovnání vzorových aplikací vyšel lépe Zend. Velké množství bodů ztratil díky své rychlosti, pokud by byly ukládána data do mezipaměti a využil se akcelerátor dosáhlo by se přijatelné rychlosti. Oproti CodeIgniter nabízí dvakrát více knihoven.

Tab. 7.8: Výhody a Nevýhody frameworků

Zend	
Výhody	**Nevýhody**
Správa uživatelů	Rychlost
Přijímání a odesílání mailů	Složitější zápis validace
Robustní knihovny	
Dostupná literatura v češtině	
Kvalitní dokumentace	
Velká komunita	
CodeIgniter	
Výhody	**Nevýhody**
Rychlost	Žádná literatura v češtin
Kvalitní dokumentace od výrobce	Žádná autentizace
	Žádná autorizace

8 SROVNÁNÍ SLOŽITOSTI NÁVRHU APLIKACÍ

Uživatel, který se rozhodne pracovat se Zend nebo s CodeIgniter, by měl mít aspoň základní znalosti psaní php. Dále by měl umět pracovat s databází, kterou zvolený framework podporuje. Pokud si tyto dvě znalosti uživatel osvojil, může se začít učit zvolený framework. Možností pro naučení frameworků je více.

Při učení CodeIgniter jsem využil video tutoriálu, který byl ve slovenštině. Dodatečně jsem si doplnil potřebné znalosti z uživatelské příručky CodeIgniter, ve která jsou daná témata dobře popsána. Po seznámení se základními vlastnostmi může programátor začít tvořit svoji aplikaci.

Zend jsem se učil z knížky *Zend Framework - programujeme webové aplikace v PHP*[2]. Při použití knížky doporučuji si vypracovat vzorovou aplikaci dle návodu. Knížka je psána pro mírně pokročilé a některé popisy nejsou brány do úplných detailů. Při nepochopení vysvětlení z knížky je možné nahlédnout do uživatelské příručky Zend, nebo vyhledat nápovědu ve fórech. Pro Zend existují česká fórum. U obou aplikací pro plynulé programování je potřeba si zvolený nástroj zažít.

Vytvoření vzorové aplikace u CodeIgniter mi trvalo tři týdny, kdy jsem každý den věnoval práci přibližně 4 hodiny. Dost času při vývoji mi zabrala samotná představa jak by měla vzorová aplikace vypadat. Návrh aplikace v Zend mi zabrala poloviční dobu. Struktura obou aplikací je shodná, při vývoji druhá aplikace jsem měl již vše naplánované. Nějaký čas si vzalo upravení aplikací aby pracovali se společnou databází. Při vývoji docházelo ke změnám, které jsem musel implementovat i u CodeIgniter. Chybou bylo, že jsem neměl naplánované přesné požadavky, co budou aplikace umět. U obou aplikací bylo nejsložitější provázat uživatelská oprávnění se všemi stránkami.

K práci se zvoleným frameworkem je potřeba nastudovat jak vytvořit základní struktury aplikace. Seznámit se s dostupnýma knihovnami, dle potřeby je implementovat. Práce s frameworky je dle mého názoru snadné. Pokud k zadanému problému framework nabídne knihovnu, usnadní tím spoustu práce. Práce v obou prostředí je stejně náročná, základní struktura obou frameworků je podobná.

9 ZÁVĚR

V této diplomové práci jsem se zabýval porovnáním dvou frameworků. Porovnával jsem rozsah nabídnutých knihoven a porovnání vlastností zjištěných při vývoji vzorové aplikace. Seznámit se s dostupnými frameworky, navrhnout vzorové aplikace a porovnat.

V první části jsem se soustředil na vyhledání dostupných frameworků v jazyce PHP, na stručný popis vlastností a na popis návrhového vzoru MVC.

Ve druhé části jsem se zaměřil na podrobné popsání knihoven zvolených frameworků a jejich funkcí. Knihovny v samotných zdrojových kódech byly stručně okomentovány pouze v hlavičkách funkcí. Popisy funkcí knihoven byly dostatečně provedeny v uživatelských manuálech od obou frameworků. Vyhledal jsem volně dostupné knihovny. Vypsal jsem knihovny použité ke zprovoznění zadaných požadavků na vzorovou aplikaci.

Ve třetí části jsem se zaměřil na nabídku možností od zvolených frameworků, při řešení vybraných problémů. Porovnal jsem dostupnost dokumentace, popisů knihoven, zda nabízí průvodce pro začínající v daném prostředí.

Ve čtvrtém bodě této práce jsem popsal navržené vzorové aplikace v prostředí Zend a CodeIgniter. Zvolené aplikace jsou popsány po částech dle zadání. Pokud to zvolený nástroj nabízel, využil jsem funkcí předpřipravených v knihovnách frameworku. U CodeIgniter jsem implementoval knihovnu pro správu uživatelů, protože základní balíček nenabízí žádnou knihovnu pro správu uživatelů a rolí. Vzorové programy obsahují všechny požadované náležitosti.

V pátém bodě této práce ukazuji navigaci mezi obrazovkami vzorových aplikací. Ukazuji ukázky některých obrazovek aplikace.

V šestém bodě popisuji užití životních cyklů u zvolených entit. V samotné práci v kapitole 6 je zobrazen diagram životních cyklů, popis stavů a možnosti přechodů.

V sedmém bodě porovnávám vzorové aplikace. Srovnáním vzorových aplikací jsem dospěl k výsledku, že Zend nabídne uživateli více ulehčení práce než CodeIgniter. Přestože je jeho rychlost několikanásobně menší než CodeIgniter, dosáhl dle mého subjektivního porovnání lepšíhch výsledku.

V poslední části této práce jsem zhodnotil časovou náročnost při vývoji aplikací a složitost na naučení.

LITERATURA

[1] LAHVIČKA, Jiří *PHP - základní informace* INTERVAL.CZ [online]. 2000. [cit. 10. 5. 2013]. Dostupné WWW:<http://interval.cz/clanky/php-zakladni-informace/>

[2] BÖHMER, Marian *Zend Framework - programujeme webové aplikace v PHP* Brno: Computer Press, 2010. 416 s. ISBN 978-80-251-2965-4.

[3] WILLIAMS, Hugh E. a David LANE. *Programujeme webové aplikace pomocí PHP a MySQL.*Vyd. 1. Praha: Computer Press, 2002. 530 s. ISBN 80-7226-760-4.

[4] ŠKRÁŠEK, Jan *PHP frameworky* PROGRAMUJEME.COM [online]. 2008. [cit. 10. 5. 2013]. Dostupné WWW: <http://programujte.com/clanek/2008022000phpframeworky/>

[5] ZEND FRAMEWORK *Zend framewor* [online]. 2013. [cit. 10. 5. 2013]. Dostupné WWW: <http://framework.zend.com/>

[6] ŠKRÁŠEK, Jan *CakePHP - začínáme s frameworkem* PROGRAMUJEME.COM [online]. 2007. [cit. 10. 5. 2013]. Dostupné WWW: <http://programujte.com/clanek/2007061601cakephpzacinames-frameworkem/>

[7] SYMBIO Digital *Symfony* SYMBIO Digital, s.r.o. [online]. 1999. [cit. 10. 5. 2013]. Dostupné z WWW: <http://www.symbio.cz/slovnik/symfony.html>

[8] RUBY ON RAILS.CZ *Začínáme s Rails* Ruby On Rails.cz [online]. 1999. [cit. 10. 5. 2013]. Dostupné z WWW: <http://guides.rubyonrails.cz/>

[9] *NETTE FRAMEWORK* nette framework [online]. 2008. [cit. 10. 5. 2013]. Dostupné z WWW: <http://nette.org/cs/hlavniprednosti>

[10] HOTOVEC, Michal *CodeIgniter — jednoduchý, efektivní framework* Hotovec [online]. 2008. [cit. 10. 5. 2013]. Dostupné WWW: <http://www.hotovec.eu/clanek/software/codeigniterjednoduchyefektivni-framework>

[11] HULÁN, Radek *Zend's PHP 5 Coding Contest - PRADO framework* INTERVAL.CZ [online]. 2004. [cit. 10. 5. 2013]. Dostupné WWW: <http://interval.cz/clanky/zendsphp5codingcontestpradoframework/>

[12] DANĚK, Petr *Velký test PHP frameworků* ROOT.CZ [online]. 2008. [cit. 10. 5. 2013]. Dostupné WWW: <http://www.root.cz/clanky/velkytestphp-frameworku2008/>

[13] *CAKE PHP* Cake PHP [online]. 2012. [cit. 10. 5. 2013]. Dostupné z WWW: <http://book.cakephp.org/>

[14] *CODEIGNITER* CodeIgniter [online]. 2001. [cit. 10. 5. 2013]]. Dostupné z WWW: <http://ellislab.com/codeigniter>

[15] *PRADO* Prado Group. [online]. 2004. [cit. 10. 5. 2013]. Dostupné z WWW: <http://www.pradoframework.com/>

[16] *JELIX* Jelix [online]. 2006. [cit. 10. 5. 2013]. Dostupné z WWW: <http://jelix.org/>

[17] BARTÁČEK, Jiří *Stránky o elektronice a počítačích* [online]. 2012. [cit. 30.12. 2012]. Dostupné WWW:<http://www.barts.cz/index.php/joomla/instalace/30-xampp>

[18] FIALA, Jan *Textový editor PSPad* [online]. 2012. [cit. 30.12. 2012]. Dostupné WWW:<http://www.pspad.com/cz/>

[19] KONYUKHOV, Ilya *Tank Auth* [online]. 2011. [cit. 30.12. 2012]. Dostupné WWW:<http://konyukhov.com/soft/tank_auth/>

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

PHP Personal Home Page - Hypertextový preprocesor

SQL Structured Query Language - Strukturovaný dotazovací jazyk

ZF Zend Framework

MSSQL Microsoft SQL Server - Relační databázový systém

MVC Model View Controller - Softwarová architektura

ACL Access Control List - Seznam řízení přístupu

API Application Programming Interface - Programovací rozhraní

HTTP Hypertext Transfer Protocol - Hypertextový internetový protokol

MIME Multipurpose Internet Mail Extensions - Víceúčelové rozšíření internetové pošty

SMTP Simple Mail Transfer Protocol - Internetový protokol

PDF Portable Document Format - Přenosový formát dokumentů

XML Extensible Markup Language - Rozšiřitelný značkovací jazyk

URL Uniform Resource Locator - Jednotný lokátor zdrojů

FTP File Transfer Protocol - Protokol pro přenos souborů

SFTP SSH File Transfer Protocol - Protokol pro bezpečný přenos souborů

LDAP Lightweight Directory Access Protocol - Protokol pro ukládání a přístup k datům

AJAX Asynchronous JavaScript and XML - Technologie vývoje interaktivních webových aplikací

XUL User Interface Language - Formát pro tvorbu multiplatformního grafického rozhraní

URL Uniform Resource Locator - Jednotný formát adres

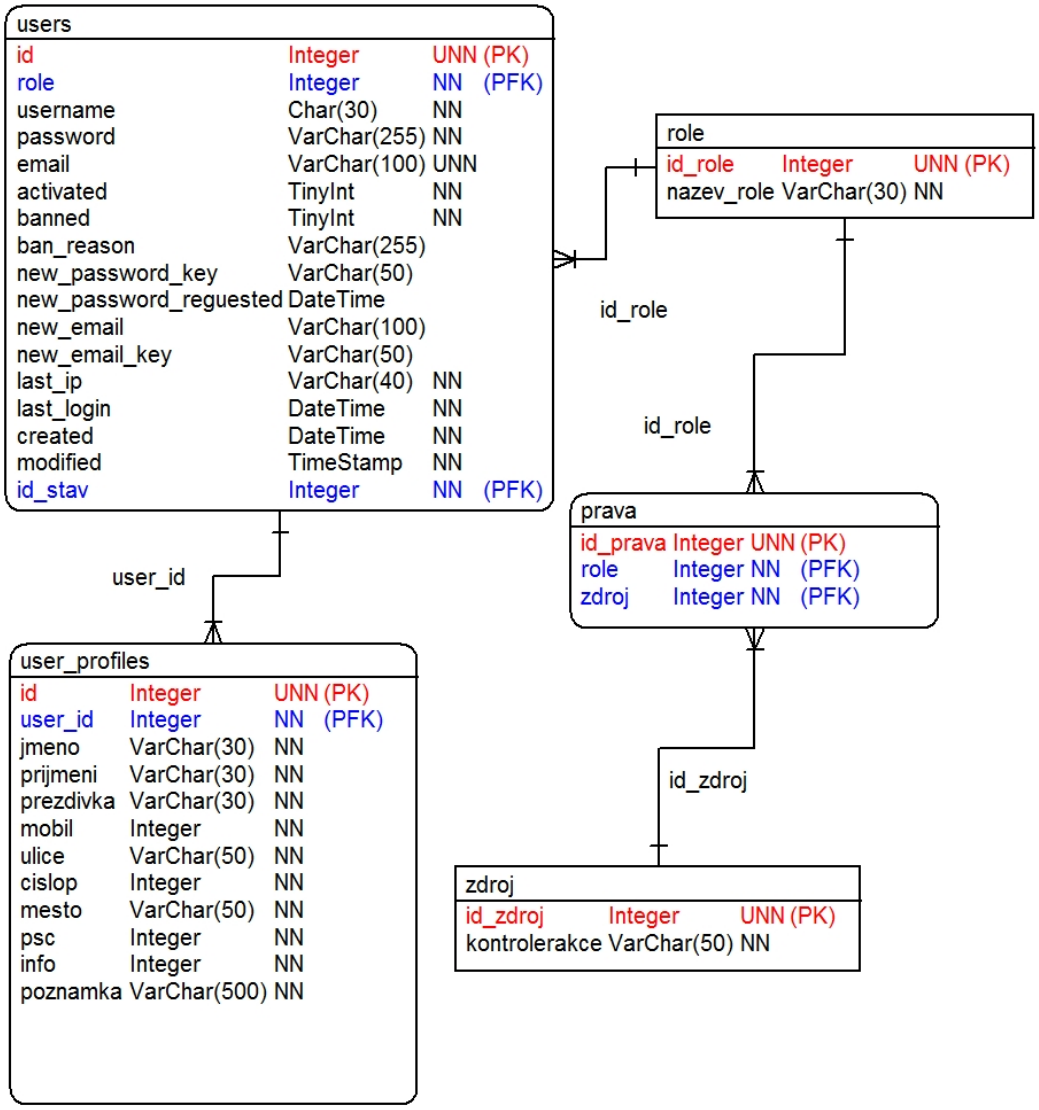
ZIP Formát pro kompresi a archivaci dat

SEZNAM PŘÍLOH

Přílohy na CD:

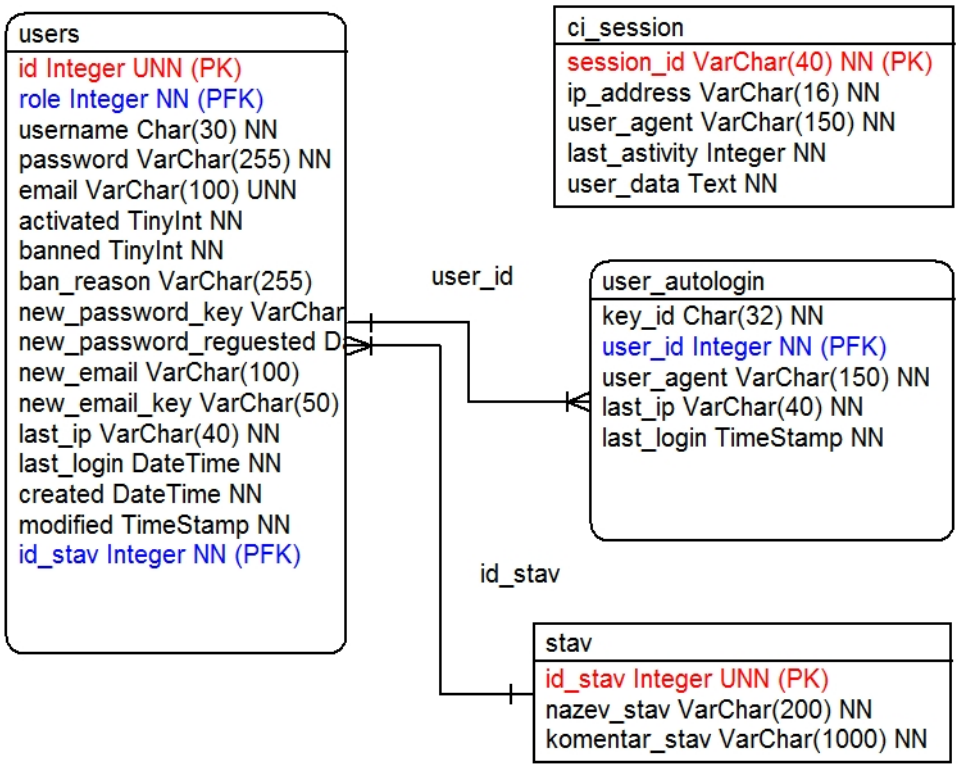
- Diplomová práce ve formátu pdf
- Složka se vzorovým kódem v programu Zend
- Složka se vzorovým kódem v programu CodeIgniter
- Složka s Databází ve formátu sql
- Složka s Obrazovkami vzorové aplikace

A DATABÁZE - UŽIVATELE



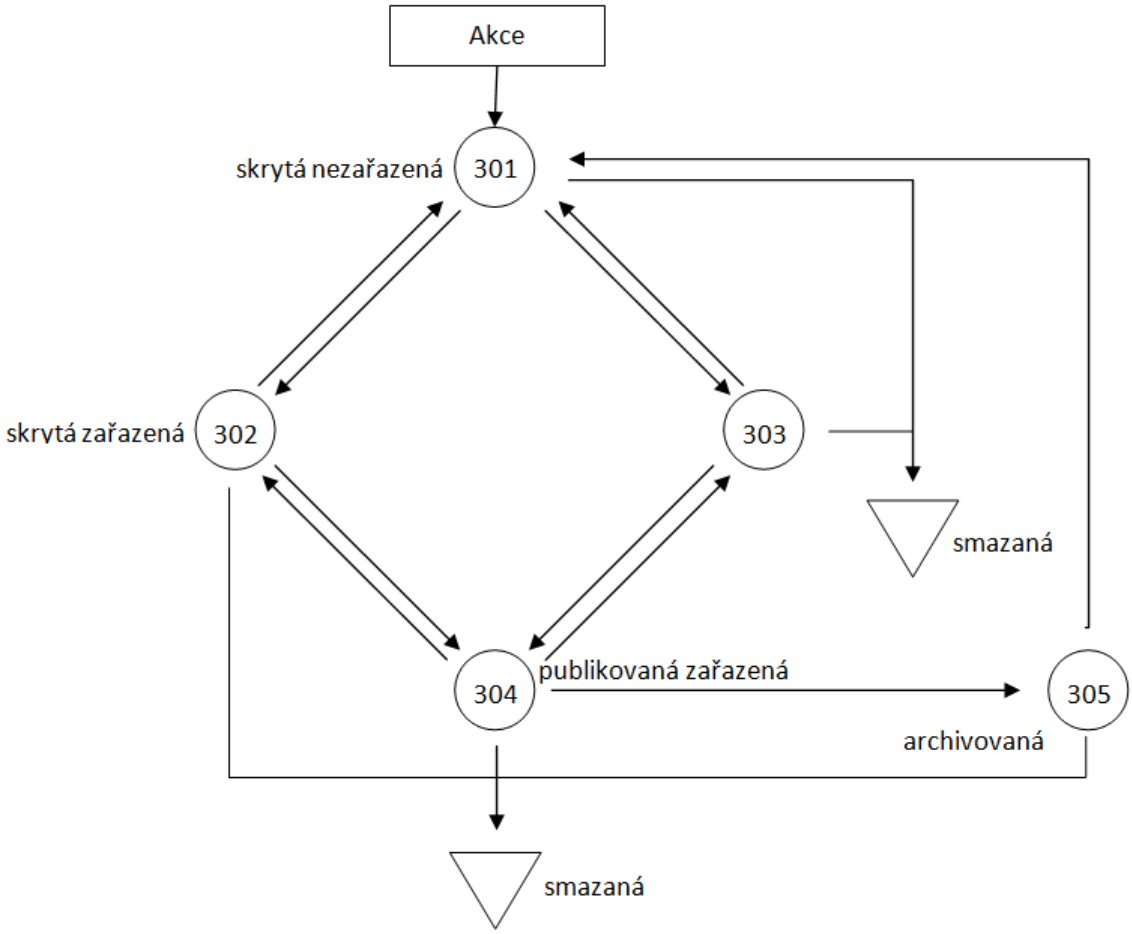
Obr. A.1: ER diagram uživatelů

B DATABÁZE - AUTENTIZACE CI



Obr. B.1: ER diagram autentizace CI

C ŽIVOTNÍ CYKLUS - AKCE



Obr. C.1: Životní cyklus Akce

Tab. C.1: Tabulka stavů Akce

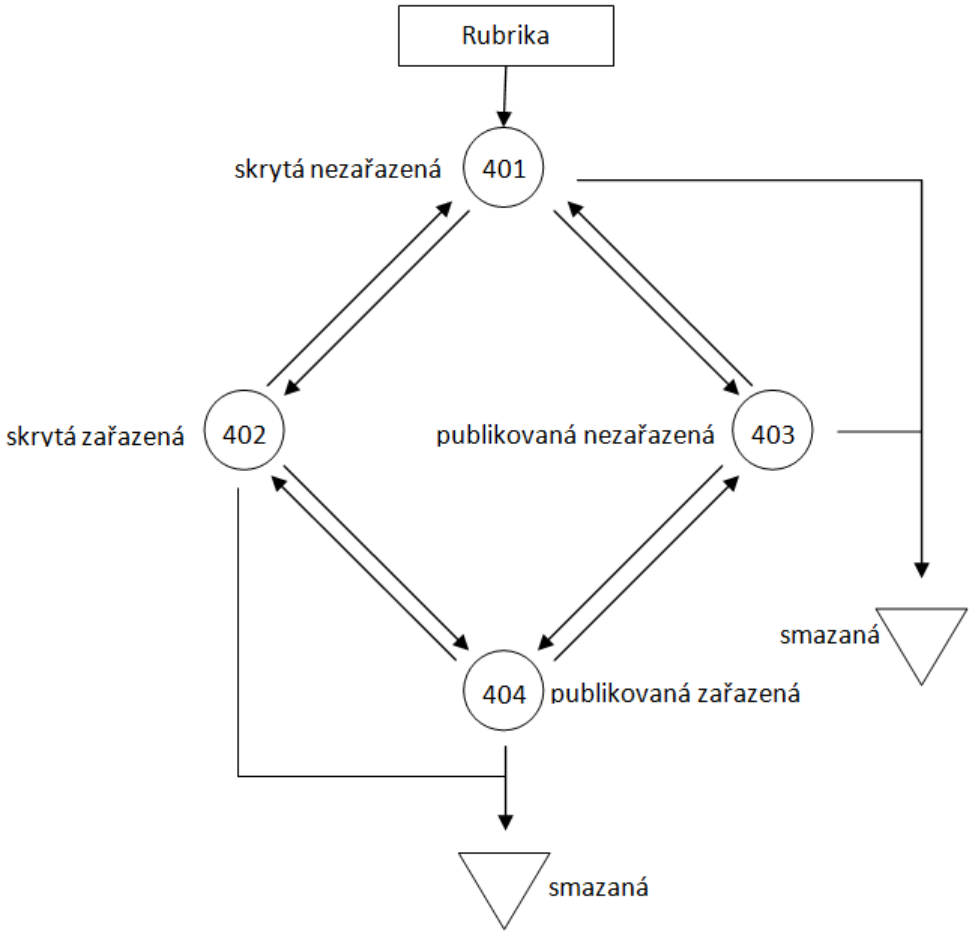
Id_stav	**Název**	**Zobrazená**
301	Akce skrytá nezařazená	NE
302	Akce skrytá zařazená	NE
303	Akce publikovaná nezařazená	NE
304	Akce publikovaná zařazená	ANO
305	Akce archivovaná	NE

„Zobrazená“ - Ano znamená, zda je stránka přístupná v levém menu, které vidí všichni uživatelé i ti nepřihlášení.

Tab. C.2: Tabulka přechodů Akce

ID_stav		**Popis přechodu**	**Role**		
Z	**DO**		**Vedouci**	**Hl.vedouci**	**Admin**
	301	Vytvoření	ANO	ANO	ANO
301	302	Zařazení do rubriky	ANO	ANO	ANO
301	303	Publikování	NE	ANO	ANO
302	301	Vyřazení z rubriky	ANO	ANO	ANO
302	303	Publikování	NE	ANO	ANO
303	304	Zařazení do rubriky	ANO	ANO	ANO
303	301	Skrytí	ANO	ANO	ANO
304	302	Skrytí	ANO	ANO	ANO
304	303	Vyřazení z rubriky	ANO	ANO	ANO
304	305	Archivování	ANO	ANO	ANO
305	301	Obnovit	ANO	ANO	ANO
301		Smazat	NE	ANO	ANO
302		Smazat	NE	ANO	ANO
303		Smazat	NE	ANO	ANO
304		Smazat	NE	ANO	ANO
305		Smazat	NE	ANO	ANO

D ŽIVOTNÍ CYKLUS - RUBRIKA



Obr. D.1: Životní cyklus Rubrika

Tab. D.1: Tabulka stavů Rubrika

Id_stav	**Název**	**Zobrazená**
401	Rubrika skrytá nezařazená	NE
402	Rubrika skrytá zařazená	NE
403	Rubrika publikovaná nezařazená	NE
404	Rubrika publikovaná zařazená	ANO

„Zobrazená“ - Ano znamená, zda je stránka přístupná v levém menu, které vidí všichni uživatelé i ti nepřihlášení.

Tab. D.2: Tabulka přechodů Rubrika

ID_stav		**Popis přechodu**	**Role**		
Z	**DO**		**Vedouci**	**Hl.vedouci**	**Admin**
	401	Vytvoření	ANO	ANO	ANO
401	402	Zařazení do menu	ANO	ANO	ANO
401	403	Publikování	NE	ANO	ANO
402	401	Vyřazení z menu	ANO	ANO	ANO
402	403	Publikování	NE	ANO	ANO
403	404	Zařazení do menu	ANO	ANO	ANO
403	401	Skrytí	ANO	ANO	ANO
404	402	Skrytí	ANO	ANO	ANO
404	403	Vyřazení z menu	ANO	ANO	ANO
401		Smazat	NE	ANO	ANO
402		Smazat	NE	ANO	ANO
403		Smazat	NE	ANO	ANO
404		Smazat	NE	ANO	ANO

E MĚŘENÍ RYCHLOSTI - VYHLEDÁVÁNÍ

Tab. E.1: Měření rychlosti - vyhledávání

Číslo měření	**Zend** [s]	**CodeIgniter** [s]
1	1,450	0,176
2	1,430	0,182
3	1,430	0,171
4	1,420	0,177
5	1,480	0,161
6	1,460	0,162
7	1,460	0,152
8	1,450	0,169
9	1,460	0,169
10	1,430	0,143
MIN	1,420	0,152
MAX	1,480	0,182
Průměr	**1,447**	**0,169**
Procentuální rozdíl		
- MIN	100 %	10,70 %
- MAX	100 %	12,30 %
- Průměr	100 %	**11,66 %**

F MĚŘENÍ RYCHLOSTI - NAČTENÍ ÚVODNÍ STRÁNKY

Tab. F.1: Měření rychlosti - načtení úvodní stránky

Číslo měření	**Zend** [s]	**CodeIgniter** [s]
1	1,380	0,181
2	1,460	0,178
3	1,410	0,170
4	1,420	0,184
5	1,380	0,188
6	1,510	0,198
7	1,350	0,189
8	1,400	0,167
9	1,320	0,188
10	1,520	0,177
MIN	1,320	0,167
MAX	1,520	0,198
Průměr	**1,415**	**0,182**
Procentuální rozdíl		
- MIN	100 %	12,65 %
- MAX	100 %	13,03 %
- Průměr	100 %	**12,86 %**

G MĚŘENÍ RYCHLOSTI - PŘIHLÁŠENÍ

Tab. G.1: Měření rychlosti - přihlášení uživatele

Číslo měření	**Zend** [s]	**CodeIgniter** [s]
1	1,310	0,152
2	1,320	0,169
3	1,310	0,151
4	1,300	0,133
5	1,320	0,116
6	1,350	0,147
7	1,330	0,122
8	1,300	0,140
9	1,340	0,134
10	1,320	0,143
MIN	1,300	0,116
MAX	1,350	0,169
Průměr	**1,320**	**0,141**
Procentuální rozdíl		
- MIN	100 %	8,92 %
- MAX	100 %	12,52 %
- Průměr	100 %	**10,66 %**

H MĚŘENÍ RYCHLOSTI - PUBLIKOVÁNÍ

Tab. H.1: Měření rychlosti - publikování stránky

Číslo měření	**Zend** [s]	**CodeIgniter** [s]
1	1,350	0,118
2	1,300	0,112
3	1,250	0,124
4	1,290	0,136
5	1,290	0,116
6	1,280	0,127
7	1,280	0,130
8	1,320	0,112
9	1,280	0,113
10	1,320	0,124
MIN	1,250	0,112
MAX	1,350	0,136
Průměr	**1,296**	**0,121**
Procentuální rozdíl		
- MIN	100 %	8,96 %
- MAX	100 %	10,07 %
- Průměr	100 %	**9,35 %**

I MĚŘENÍ RYCHLOSTI - OTEVŘENÍ RUBRIKY V MENU

Tab. I.1: Měření rychlosti - otevření rubriky v menu

Číslo měření	**Zend** [s]	**CodeIgniter** [s]
1	1,350	0,171
2	1,340	0,170
3	1,320	0,161
4	1,360	0,174
5	1,330	0,161
6	1,290	0,156
7	1,330	0,173
8	1,330	0,148
9	1,310	0,152
10	1,340	0,144
MIN	1,290	0,144
MAX	1,360	0,174
Průměr	**1,330**	**0,161**
Procentuální rozdíl		
- MIN	100 %	11,16 %
- MAX	100 %	12,79 %
- Průměr	100 %	**12,11 %**